File No. S360-32
Form C28-6586-2

OS

IBM Systems Reference Library

IBM System/360 Operating System

Utilities

This publication discusses the capabilities of the IBM System/360 Operating System utility programs and the control statements used with each program. These programs are used by programmers responsible for organizing and maintaining operating system data.

Three types of utility programs are discussed: system utilities and data set utilities, which are used directly with the System/360 Operating System; and independent utilities, which operate outside the operating system. System utilities deal with operating system control data. Data set utilities manipulate data sets at the record level and above. Independent utilities initialize, dump, and restore direct-access volumes.

PREFACE

System/360 Operating System utility programs provide functions to assist programmers responsible for creating and maintaining operating system data. These functions are requested through simple control statements, which can be used individually or in combination, to perform a variety of housekeeping and support operations.

This publication discusses the functions provided by each utility program and the control statements used to request these functions. Appendixes A and B contain information for linking to exit routines and invoking utility programs. Appendixes C, D, and E contain the error messages and return codes issued by the utility programs.

The reader should be familiar with the concepts and terminology introduced in the prerequisite publications.

PREREQUISITE PUBLICATIONS

IBM System/360 Operating System: Concepts and Facilities, Form C28-6535

IBM System/360 Operating System: Job Control Language, Form C28-6539

IBM System/360 Operating System: Data Management, Form C28-6537

Third Edition

This is a reprint of C28-6586-2 incorporating changes released in TNL N28-2133. Significant changes or additions to the specifications will be reported in subsequent revisions or Technical Newsletters.

This publication was prepared for production using an IBM computer to update the text and to control the page and line format. Page impressions for photo-offset printing were obtained from an IBM 1403 Printer using a special print chain.

Copies of this and other IBM publications can be obtained through IBM Branch Offices.

A form for readers' comments appears at the back of this publication. It may be mailed directly to IBM. Address any additional comments concerning this publication to the IBM Corporation, Programming Systems Publications, Department D58, PO Box 390, Poughkeepsie, N. Y. 12602

FIGURES

TABLES

The IBM System/360 Operating System provides utility programs to assist in organizing and maintaining data. There are three types of programs: system utilities, data set utilities, and independent utilities.

System utility programs, executed under the operating system:

- Modify system control data by building and maintaining catalogs and volume structures.
- Move and copy collections of data in order to rearrange data and create backup copies.
- List a catalog, a directory of a partitioned data set, and a volume table of contents.

Data set utility programs, also executed under the operating system:

- Copy and merge partitioned data set members.
- Copy records from sequential data sets and convert sequential data sets to partitioned organization.
- Compare records in sequential or partitioned data sets.
- Print or punch records in sequential or partitioned data sets.
- Update symbolic library modules at the source language level.

Independent utility programs, executed outside of the System/360 Operating System:

- Initialize and assign alternate tracks to direct-access volumes.
- Dump and restore the data contents of a direct-access volume.

Format of Utility Control Statements

The control statements for the System/360 operating system utility programs have the following standard format:

Name	Operation	Operand
Optional symbolic name	Control statement type	Optional and required parameters

The name symbolically identifies the control statement and can be omitted at the discretion of the utility user. When included, a name must begin in the first position of the statement and must be followed by one or more blanks. It can contain from one to eight alphameric characters, the first of which must be alphabetic.

The operation identifies the type of control statement. It must be preceded and followed by one or more blanks.

The operand is made up of one or more keyword parameters separated by commas. The operand field is ended with a blank. Commas, parentheses, and blanks can only be used as delimiting characters.

Comments can be written in a utility statement, but they must be separated from the last parameter of the operand field by one or more blanks.

A typical utility statement might appear as:

```
NAME OPERATION KEYWORD=information,...
```

Utility statements are coded on cards or as card images and are contained in columns 1 through 71. A statement that exceeds 71 characters can be continued onto one or more additional cards; a nonblank character is placed in column 72 to indicate continuation. The statement can be interrupted either in column 71 or after any comma. The continued portion of the statement must begin in column 16 of the following card. Comments are continued in column 17 of the following card, or can appear on every card of a continued statement.

Notation for Defining Control Statements

The notation used to define control statements in this publication is described in the following paragraphs.

1. The set of symbols listed below are used to define control statements, but are never written in an actual statement.

hyphen	-
underscore	_
braces	{}
brackets	[]
ellipsis	...

The special uses of these symbols are explained in paragraphs 5-9.

2. Upper-case letters and words, numbers, and the set of symbols listed below are written in an actual control statement exactly as shown in the statement definition.

apostrophe	'
asterisk	*
comma	,
equal sign	=
parentheses	()
period	.

3. Lower-case letters, words, and symbols appearing in a control statement definition represent variables for which specific information is substituted in the actual statement.

Example: If name appears in a statement definition, a specific value (e.g., ALPHA) is substituted for the variable in the actual statement.

4. Stacked items represent alternatives. Only one such alternative should be selected.

Example: The representation

A
B
C

indicates that either A or B or C should be selected.

5. Hyphens join lower-case letters, words, and symbols to form a single variable.

Example: If member-name appears in a statement definition, a specific value (e.g., BETA) is substituted for the variable in the actual statement.

6. An underscore indicates a default option. If an underscored alternative is selected, it need not be written in the actual statement.

Example: The representation

$$\left\{\begin{matrix} A \\ \underline{B} \\ C \end{matrix}\right\}$$

indicates that either A or B or C should be selected; however, if B is selected, it need not be written, because it is the default option.

7. Braces group related items, such as alternatives.

Example: The representation

$$ALPHA=(\left\{\begin{matrix} A \\ \underline{B} \\ C \end{matrix}\right\},D)$$

indicates that a choice should be made among the items enclosed within the braces. If A is selected, the result is ALPHA=(A,D). If C is selected, the result can be either ALPHA=(,D) or ALPHA=(C,D).

8. Brackets also group related items; however, everything within the brackets is optional and may be omitted.

Example: The representation

$$ALPHA=(\left[\begin{matrix} A \\ B \\ C \end{matrix}\right],D)$$

indicates that a choice can be made among the items enclosed within the brackets or that the items within the brackets can be omitted. If B is selected, the result is: ALPHA=(B,D). If no choice is made, the result is: ALPHA=(,D).

9. An ellipsis indicates that the preceding item or group of items can be repeated more than once in succession.

Example: ALPHA[,BETA]... indicates that ALPHA can appear alone or can be followed by ,BETA any number of times in succession.

System utility programs manipulate collections of data and system control information. They include:

- IEHPROGM -- a program that builds and maintains system control data.
- IEHMOVE -- a program that moves and copies collections of data.
- IEHLIST -- a program that lists system control data.

This section describes the capabilities, requirements for execution, and examples of the use of each system utility program.

Job Control Statement Requirements

System utility programs can be introduced as jobs or job steps, and are executed in response to job control statements and utility control statements. The job control statements used are:

- A JOB statement.
- An EXEC statement that identifies the system utility program to be executed.
- A DD statement named SYSPRINT that defines an output data set used by the utility program.
- A group of DD statements that allocate the devices required by the utility program.
- A DD statement named SYSIN that defines a sequential data set used for utility statement input.

Figure 1 illustrates a set of job control statements used to execute a system utility program.

System utility programs can also be employed as subroutines by use of the System/360 Operating System linkage conventions. At the completion or termination of the utility program, the highest return code encountered within the program is passed to the calling program. Invocation of utility programs and linkage conventions are discussed in Appendix B.

Cataloging Procedures

Execution of a system utility program can be simplified by cataloging a set of job control statements for each program. The result is a cataloged procedure. The DD statements in a procedure should ensure that a sufficient number of devices are allocated for most applications of the program. The cataloged DD statements can be overridden or modified for applications that require more devices.

A set of control statements might include:

- An EXEC statement that identifies the utility program to be executed.
- A DD statement named SYSPRINT that defines a sequential data set used for message output.
- One DD statement for each permanently mounted volume. (The system residence volume is considered to be permanent.)
- One DD statement for each type of mountable device.
- One DD statement for each additional mountable device required by the program.

Note: DD statements defining permanently mounted devices must specify DISP=(OLD,...) to prevent deletion of the data set. DD statements defining mountable devices must specify UNIT=(...,DEFER), DISP=(NEW,KEEP), and VOLUME=(PRIVATE,...).

Executing Cataloged Procedures

The job control statements required to execute a cataloged procedure are:

- A JOB statement.
- An EXEC statement that specifies PROC=procname, where procname is the name of the cataloged procedure.
- A group of DD statements that define the additional device requirements of a particular job, or that modify cataloged DD statements.

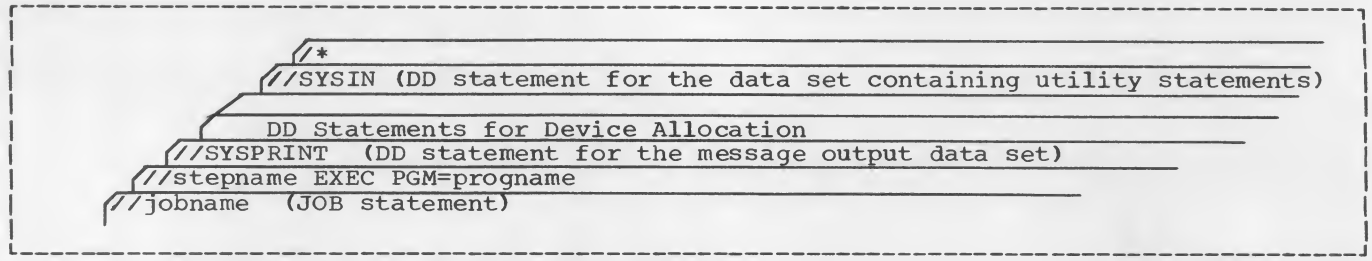

Figure 1. Executing a System Utility Program

Figure 2 illustrates job control statements and utility statements used together to execute a cataloged procedure.

Identifying Volumes and Data Sets

A utility statement identifies the particular function to be performed and, when required, the specific volumes or data sets to be used. Volumes are identified by device type and volume serial number.

A data set residing on a direct-access volume is identified by the volume serial number and its data set name; a data set residing on a tape volume must be further identified by its data set sequence number. In general, identification parameters are coded as follows:

```
KEYWORD=device=(serial,seqno,...)
```

Applicable keywords include VOL, CVOL, FROM, and TO.

The term "device" is replaced by any of a group of device names that the installation defines when the operating system is generated. Each device name may represent:

- A single device.
- All devices of a specific type.

Device names that represent two or more different device types cannot be used.

The term "serial" is replaced by a 1- to 6-character volume serial number.

The term "seqno" is replaced by a data set sequence number. When a device other than tape is specified, the sequence number is omitted, as follows:

```
KEYWORD=device=(serial,serial,...)
```

If only a single direct-access device is required, the parentheses can be deleted, as follows:

```
KEYWORD=device=serial
```

Hereafter, a volume parameter that may require more than one volume will be referred to as:

```
KEYWORD=device=list
```

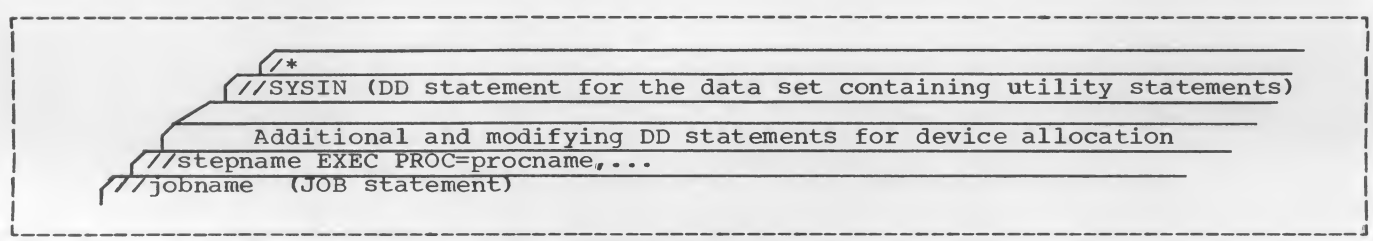

Figure 2. Executing a Cataloged Utility Procedure

MODIFYING SYSTEM CONTROL DATA

The IEHPROGM utility program provides efficient facilities for modifying system control data. It can be used to scratch, rename, catalog, and uncatalog data sets; to build and delete indexes or index aliases; to connect and release control volumes; and to build generation indexes. This program is executed or invoked with the symbolic name IEHPROGM.

Scratching a Data Set

The SCRATCH statement is used to scratch a data set or member from a direct-access volume. A data set is considered scratched when its data set label is removed from the volume table of contents of the volume on which it resides, and its space is made available for reallocation. A member is considered scratched when its name is removed from the directory of the partitioned data set in which it is contained. A data set or member is scratched only from the volumes designated in the SCRATCH statement. This function does not uncatalog scratched data sets.

Note: If the device operates in split-cylinder mode, the tracks occupied by a scratched data set will not be available for reallocation until all data sets sharing the cylinder have been scratched.

Note: A password-protected data set being scratched is over-written to prevent retrieval of its contents.

Name	Operation	Operand
[name]	SCRATCH	{DSNAME=name / VTOC}
		VOL=device=list
		[PURGE]
		[MEMBER=name]
		[SYS]

DSNAME=name
: specifies the fully qualified name of either the data set to be scratched, or the partitioned data set that contains the member to be scratched.

VTOC
: specifies that all data sets on the specified volume, except those protected by a password, are to be scratched. Password-protected data sets are scratched if the correct password is provided.

VOL=device=list
: specifies the volume or volumes that contain the data set or data sets to be scratched. If VTOC is specified, VOL should not specify a system residence volume.

PURGE
: requests that each data set specified by DSNAME or VTOC be scratched, even if its expiration date has not elapsed.

  If PURGE is omitted, the specified data sets are scratched only if their expiration dates have elapsed.

MEMBER=name
: specifies the name of a member to be scratched from the named data set.

  If omitted, the entire data set or volume of data sets is scratched.

SYS (used only with SCRATCH VTOC operations)
: requests that data sets whose names begin with AAAAAAAA.AAAAAAAA.AAAAAAAA.AAAAAAAA be scratched. (These are names assigned to data sets by the operating system.)

Renaming a Data Set (RENAME)

The RENAME statement is used to change the name of a data set or member that resides on a direct-access volume. The data set or member is renamed only on the designated volumes. This function does not update the catalog after the operation.

Name	Operation	Operand
[name]	RENAME	DSNAME=name
		VOL=device=list
		NEWNAME=name
		[MEMBER=name]

DSNAME=name
: specifies the fully qualified name of the data set to be renamed, or the data set that contains the member to be renamed.

VOL=device=list
: specifies the volume or volumes that contain the data set to be renamed.

NEWNAME=name
: specifies the new fully qualified name for the data set, or the new member name.

MEMBER=name
: specifies the name of a member (in the named data set) that is to be renamed.

  If omitted, the entire data set is renamed.

Cataloging a Data Set (CATLG)

The CATLG statement is used to generate an entry in an index of the catalog. The entry contains a data set name and associated volume information. If higher level indexes are required, the CATLG function automatically creates them.

Name	Operation	Operand
[name]	CATLG	DSNAME=name VOL=device=list [CVOL=device=serial]

DSNAME=name
specifies the fully qualified name of the data set to be cataloged.

VOL=device=list
specifies the volume or volumes that contain the data set to be cataloged.

CVOL=device=serial
specifies the device type and volume serial number of the control volume at which the catalog search for the index is to begin.

If CVOL is omitted, the system residence volume is assumed.

Uncataloging a Data Set (UNCATLG)

The UNCATLG statement is used to remove an entry from an index of the catalog.

Name	Operation	Operand
[name]	UNCATLG	DSNAME=name [CVOL=device=serial]

DSNAME=name
specifies the fully qualified name of the data set to be uncataloged.

CVOL=device=serial
specifies the device type and volume serial number of the control volume at which the search for the catalog entry is to begin.

If CVOL is omitted, the system residence volume is assumed.

Building an Index (BLDX)

The BLDX statement is used to create a new index in the catalog. If the creation of an index requires that higher level indexes be created, the BLDX function automatically creates them.

Name	Operation	Operand
[name]	BLDX	INDEX=name [CVOL=device=serial]

INDEX=name
specifies the qualified name of the index to be created.

CVOL=device=serial
specifies the device type and volume serial number of the control volume at which the catalog search for the index is to begin.

If CVOL is omitted, the system residence volume is assumed.

Deleting an Index (DLTX)

The DLTX statement is used to remove an index from the catalog. Only an index that has no entries can be removed.

Name	Operation	Operand
[name]	DLTX	INDEX=name [CVOL=device=serial]

INDEX=name
specifies the qualified name of the index to be deleted.

CVOL=device=serial
specifies the device type and volume serial number of the control volume at which the catalog search for the index is to begin.

If CVOL is omitted, the system residence volume is assumed.

Building an Index Alias (BLDA)

The BLDA statement is used to assign an alternate name to an index at the highest level of the catalog.

Name	Operation	Operand
[name]	BLDA	INDEX=name ALIAS=name [CVOL=device=serial]

INDEX=name
specifies the unqualified index name to which an alias is to be assigned.

ALIAS=name
specifies an unqualified name to be assigned as the alias.

CVOL=device=serial
specifies the device type and volume serial number of the control volume where the catalog entry is to be made.

If CVOL is omitted, the system residence volume is assumed.

Deleting an Index Alias (DLTA)

The DLTA statement is used to delete an alternate name previously assigned to an index at the highest level of the catalog.

Name	Operation	Operand
[name]	DLTA	ALIAS=name [CVOL=device=serial]

ALIAS=name
specifies the unqualified index alias to be deleted.

CVOL=device=serial
specifies the device type and volume serial number of the control volume containing the catalog entry to be deleted.

If CVOL is omitted, the system residence volume is assumed.

Connecting Two Control Volumes (CONNECT)

The CONNECT statement is used to place an entry, containing an index name and a volume serial number, into the volume index of a control volume. This effectively connects two control volumes.

Name	Operation	Operand
[name]	CONNECT	INDEX=name VOL=device=serial [CVOL=device=serial]

INDEX=name
specifies the index name to be entered in the specified volume index.

VOL=device=serial
specifies the device type and volume serial number to be entered in the specified volume index.

CVOL=device=serial
specifies the device type and volume serial number of the control volume into whose volume index the entry is to be placed.

If CVOL is omitted, the system residence volume is assumed.

Disconnecting Two Control Volumes (RELEASE)

The RELEASE statement is used to remove an entry from the volume index of a control volume. This effectively disconnects two control volumes.

Name	Operand	Operand
[name]	RELEASE	INDEX=name [CVOL=device=serial]

INDEX=name
specifies the index name to be removed from the specified volume index.

CVOL=device=serial
specifies the device type and volume serial number of the control volume from whose volume index the entry is to be removed.

If CVOL is omitted, the system residence volume is assumed.

Building a Generation Data Group (BLDG)

The BLDG statement is used to build an index for a generation data group and to establish the action to be taken when the index overflows. In order to build a generation data group index, a model data set control block (DSCB) must have been created on the same control volume. A model DSCB is established with a DD statement that requests a space allocation of zero tracks.

If a model DSCB does not exist on this control volume, the function is not performed.

Name	Operation	Operand
[name]	BLDG	INDEX=name
		ENTRIES=n
		[CVOL=device=serial]
		[EMPTY]
		[DELETE]

INDEX=name
specifies the name of the generation data group index. A DSCB with the same name must reside on the same control volume.

ENTRIES=n
specifies the number of entries to be contained in the generation data group index; n must not exceed 255.

CVOL=device=serial
specifies the device type and volume serial of the control volume at which the catalog search for the generation data group index is to begin.

If CVOL is omitted, the system residence volume is assumed.

EMPTY
specifies that all entries be removed from the generation data group index when it overflows.

If EMPTY is omitted, the oldest entry will be removed when the generation data group index overflows.

DELETE
specifies that data sets are to be scratched after their entries are removed from the generation data group index.

If DELETE is omitted, the data sets are not scratched.

Example

Figure 3 illustrates some applications of the IEHPROGM utility program. GROUP1 scratches data set A.B.C and removes its entry from the A.B index. GROUP2 connects the system residence volume with control volume P2, and builds index X.Y on P2. GROUP3 changes the data set name R.S.T to C.D.E. GROUP4 builds a new generation data group index G.D.G2, and deletes an existing index G.D.G1 whose entries have been removed with a series of UNCATLG operations.

Sample Coding Form

```
//JOB1     JOB    09-550,BROWN
//        EXEC   PGM=IEHPROGM
//SYSPRINT DD     SYSOUT=A
//DD1      DD     VOLUME=SER=1234,DISP=OLD,UNIT=2302
//DD2      DD     VOLUME=PRIVATE,UNIT=(2311,,DEFER),DISP=(NEW,KEEP)
//DD3      DD     VOLUME=PRIVATE,UNIT=(2400-2,,DEFER),DISP=(NEW,KEEP)
//SYSIN    DD     *
GROUP1   SCRATCH     DSNAME=A.B.C,VOL=2311=P3
         UNCATLG     DSNAME=A.B.C
GROUP2   CONNECT     INDEX=X,VOL=2311=P2
         BLDX        INDEX=X.Y,CVOL=2311=P2
GROUP3   RENAME      DSNAME=R.S.T,VOL=2302=D1,NEWNAME=C.D.E
GROUP4   BLDG        INDEX=G.D.G2,ENTRIES=5,DELETE
         DLTX        INDEX=G.D.G1
/*
```

Figure 3. *Modifying* System Control Data

MOVING AND COPYING DATA

The IEHMOVE utility program moves and copies logical collections of IBM System/360 Operating System data. The program can collect fragments of a data set onto one volume, clear an entire volume for reassignment, copy data sets, reorder a collection of data, and perform a variety of other housekeeping operations. This program is executed or invoked with the symbolic name IEHMOVE.

A moving operation relocates a collection of data and scratches the source data. A copying operation produces a replica of the source collection and leaves the source data intact. IEHMOVE can perform move and copy operations on:

- A data set.
- A group of data sets.
- A partitioned data set.
- A catalog.
- A volume.

A partitioned data set can be moved or copied as if it were an ordinary data set, or as a partitioned data set to take advantage of a number of additional operations. (See Table 1.) Any number of operations can be performed within a given job. If the data is movable, space is allocated automatically, if required. If the data is unmovable, space must be preallocated.

Cautions: An unmovable data set can be successfully moved or copied only if space has been previously allocated on its receiving volume. (A data set may have been declared "unmovable" if it contains any location-dependent information. This attribute is indicated in the organization section of the data set control block associated with the data set.)

IEHMOVE will attempt to move or copy an unmovable data set, regardless of the reasons for which it was declared unmovable. An exception is a BDAM data set which is unmovable or one that is movable for which space has been pre-allocated. (See Table 2.)

System data should be moved with care to ensure its subsequent recovery.

When moving or copying operations involve a magnetic tape, 9-track tape is assumed to have standard labels, and 7-track tape is assumed to have (1) standard labels written in low density, even parity, with translate on, and (2) data written in odd parity with translate and data conversion off.

Results of Moving and Copying Operations: A moving or copying operation has one of three results:

- A successful operation.
- No action.
- An "unloaded" version of the source data.

An unloaded version is a sequential data set that contains information for reconstructing the source data. It can be used to produce a successful operation by moving or copying it onto a volume that would have originally supported a successful operation.

The above results depend upon the compatibility of the source and receiving volumes with respect to size and data set organization, the movability of the source data set, and the allocation of space on the receiving volume. (Two volumes are compatible with respect to size if the source record size does not exceed the receiving track size.) Table 2 indicates the results of moving and copying operations for the various receiving volumes and source data sets. The organization of the source data set is noted at the top of the table, and the characteristics of the receiving device are listed in the first column of the table.

Additional Job Control Statement Requirements: This program requires a DD statement named SYSUT1 that defines a direct-access device, in addition to those DD statements described in the section "Job Control Statement Requirements." This statement defines a work data set used only with IEHMOVE.

Utility Statements: Table 1 associates the types of data collections with the MOVE and COPY utility statements. The INCLUDE, EXCLUDE, SELECT, and REPLACE utility statements can be used to enlarge or restrict the scope of certain MOVE and COPY statements.

Table 1. MOVE/COPY Utility Statements

Data Entity	Utility Statements	Additional Statements
Data Set[2]	{MOVE | COPY} DSNAME	None
Group of Data Sets[2]	{MOVE | COPY} DSGROUP	INCLUDE EXCLUDE
Partitioned Data Set	{MOVE | COPY} PDS	INCLUDE EXCLUDE REPLACE SELECT
Catalog	{MOVE | COPY} CATALOG	EXCLUDE
Volume of Data	{MOVE | COPY} VOLUME	None

[2]Including partitioned data sets.

Table 2. Results of Moving and Copying Operations

Receiving Volume	Source Data Set Organization		
	Sequential	Partitioned	Direct-Access
Non-Direct-Access			
(Movable Data)	Successful	Unloaded	Unloaded
(Unmovable Data)	Unloaded	Unloaded	Unloaded
Direct-Access (Compatible size)			
• Space allocated by utility program (Movable Data)	Successful	Successful	Successful
• Space allocated by utility program (Unmovable Data)	Unloaded	Unloaded	Unloaded[1]
• Space previously allocated, not used	Successful	Successful	Successful[1]
• Space previously allocated, partially used	No action	Successful (merge)	No action
Direct-Access (Incompatible size)			
• Space allocated by utility program	Unloaded	Unloaded	Unloaded
• Space previously allocated, not used	Unloaded	Unloaded	No action
• Space previously allocated, partially used	No action	No action	No action

[1]For BDAM data sets, no action.

Moving a Data Set (MOVE DSNAME)

The MOVE DSNAME statement is used to move a single data set within a volume or from one volume to another. If the data set is cataloged, the catalog is automatically updated unless UNCATLG is specified.

Name	Operation	Operand
[name]	MOVE	DSNAME=name TO=device=list [FROM=device=list CVOL=device=serial] [UNCATLG] [RENAME=name]

DSNAME=name
specifies the fully qualified name of the data set to be moved.

TO=device=list
specifies the volume or volumes to which the data set is to be moved.

FROM=device=list
specifies the volume or volumes on which the data set currently resides, if it is not cataloged.

If the data set is cataloged, FROM should not be written.

CVOL=device=serial
specifies the device type and volume serial number of the control volume at which the catalog search for the data set is to begin.

If neither CVOL nor FROM is written, the data set is assumed to be cataloged on the system residence volume.

UNCATLG
specifies that the catalog entry pertaining to the data set is to be removed. This parameter should be used only if the source data set is cataloged.

UNCATLG is ignored if the volume was identified by FROM.

RENAME=name
specifies that the data set is to be renamed, and indicates the new name.

Copying a Data Set (COPY DSNAME)

The COPY DSNAME statement is used to copy a data set onto another volume. The source data set, if cataloged, remains cataloged unless UNCATLG is specified. The copy of the data set will be cataloged on its receiving volume if CATLG is specified.

Name	Operation	Operand
[name]	COPY	DSNAME=name TO=device=list [FROM=device=list CVOL=device=serial] [UNCATLG] [CATLG] [RENAME=name]

DSNAME=name
specifies the fully qualified name of the data set to be copied.

TO=device=list
specifies the volume or volumes on which the data set is to be copied.

FROM=device=list
specifies the volume or volumes on which the data set currently resides, if it is not cataloged.

If the data set is cataloged, FROM should not be written.

CVOL=device=serial
specifies the device type and volume serial number of the control volume at which the catalog search for the data set is to begin.

If neither CVOL nor FROM is written, the data set is assumed to be cataloged on the system residence volume.

UNCATLG
specifies that the catalog entry pertaining to the source data set is to be removed. (The source data set must be cataloged.)

UNCATLG is ignored if the volume was identified by FROM.

CATLG
specifies that the copy of the data set is to be cataloged on the receiving volume, if it is a direct-access volume. If a catalog does not exist on the receiving volume, a new catalog is created.

RENAME=name
specifies that the data set is to be renamed, and indicates the new name.

Moving a Group of Data Sets (MOVE DSGROUP)

The MOVE DSGROUP statement is used to move groups of data sets that are cataloged on the same control volume and whose names are partially qualified by one or more identical names. For example, a group of data sets qualified by the name A.B includes data sets named A.B.C and A.B.D, but does not include data sets named A.C.D or A.D.F. MOVE DSGROUP operations cause the specified catalog to be updated automatically unless UNCATLG is specified. The INCLUDE and EXCLUDE statements, discussed later in this section, can be used to add to or delete data sets from the group.

Name	Operation	Operand
[name]	MOVE	DSGROUP[=name]
		TO=device=list
		[CVOL=device=serial]
		[PASSWORD]
		[UNCATLG]

DSGROUP=name
: specifies a qualified name. All cataloged data sets whose names are qualified by this name are to be moved. If the name is a fully qualified data set name, the operation is performed only on that data set.

DSGROUP
: specifies that all data sets cataloged on the specified control volume are to be moved.

TO=device=list
: specifies the volume or volumes to which the specified group of data sets is to be moved.

CVOL=device=serial
: specifies the device type and volume serial number of the control volume at which the catalog search for the data sets is to begin.

  If CVOL is omitted, the specified group of data sets is assumed to be cataloged on the system residence volume.

PASSWORD
: specifies that password-protected data sets contained in the group are to be considered eligible for the moving operation.

  If PASSWORD is omitted, only data sets that are not password-protected are moved.

UNCATLG
: specifies that the catalog entries pertaining to the specified group of data sets are to be removed.

Copying a Group of Data Sets (COPY DSGROUP)

The COPY DSGROUP statement is used to copy groups of data sets that are cataloged on the same control volume and whose names are partially qualified by one or more identical names. The source data sets remain cataloged unless UNCATLG is specified. The copies of the data sets are cataloged on their receiving volumes if CATLG is specified. The INCLUDE and EXCLUDE statements, discussed later in this section, can be used to add data sets to or delete data sets from the group.

Name	Operation	Operand
[name]	COPY	DSGROUP[=name] TO=device=list [CVOL=device=serial] [PASSWORD] [UNCATLG] [CATLG]

DSGROUP=name
: specifies a qualified name. All cataloged data sets whose names are qualified by this name are to be copied. If the name is a fully qualified data set name, the operation is performed only on that data set.

DSGROUP
: specifies that all data sets cataloged on the specified control volume are to be copied.

TO=device=list
: specifies the volume or volumes on which the specified group of data sets is to be copied.

CVOL=device=serial
: specifies the device type and volume serial number of the control volume at which the catalog search for the data sets is to begin.

  If CVOL is omitted, the specified group of data sets is assumed to be cataloged on the system residence volume.

PASSWORD
: specifies that password-protected data sets contained in the group are to be considered eligible for the copying operation.

  If PASSWORD is omitted, only data sets that are not password-protected are copied.

UNCATLG
: specifies that the catalog entries pertaining to the source group of data sets are to be removed.

CATLG
: specifies that the names of the copied data sets are to be cataloged on their receiving volumes, if they are direct-access volumes. If catalogs do not exist on the receiving volumes, they are created.

Moving a Partitioned Data Set (MOVE PDS)

The MOVE PDS statement is used to move partitioned data sets. In addition, it can be used to expand a directory and, in conjunction with the INCLUDE, EXCLUDE, REPLACE, and SELECT statements, to merge selected members of several partitioned data sets or delete members. MOVE PDS causes the specified catalog to be automatically updated unless UNCATLG is specified. If the receiving volume already contains a partitioned data set with the same name, the two are merged.

Name	Operation	Operand
[name]	MOVE	PDS=name TO=device=serial [FROM=device=serial] [CVOL=device=serial] [EXPAND=n] [UNCATLG] [RENAME=name]

PDS=name
: specifies the fully qualified name of the partitioned data set to be moved.

TO=device=serial
: specifies the device type and volume serial number of the volume to which the partitioned data set is to be moved.

FROM=device=serial
: specifies the device type and volume serial number of the volume on which the partitioned data set currently resides, if it is not cataloged.

  If the data set is cataloged, FROM should not be written.

CVOL=device=serial
: specifies the device type and volume serial number of the control volume at which the catalog search for the partitioned data set is to begin.

  If neither FROM nor CVOL is written, the partitioned data set is assumed to be cataloged on the system residence volume.

EXPAND=n
: specifies the number of 256-byte records to be added to the directory of the specified partitioned data set.

UNCATLG
: specifies that the catalog entry pertaining to the source partitioned data set is to be removed. This parameter should be used only if the source partitioned data set is cataloged.

  If the volume was identified by FROM, UNCATLG is ignored.

RENAME=name
: specifies that the data set is to be renamed, and indicates the new name.

Copying a Partitioned Data Set (COPY PDS)

The COPY PDS statement is used to copy partitioned data sets. In addition, it can be used to expand a directory and, in conjunction with the INCLUDE, EXCLUDE, REPLACE, and SELECT statements, to merge selected members from several partitioned data sets or delete members. If the receiving device already contains a partitioned data set having the same name as the source partitioned data set, the two data sets are merged. The source partitioned data set remains cataloged unless UNCATLG is specified.

Name	Operation	Operand
[name]	COPY	PDS=name
		TO=device=serial
		[FROM=device=serial]
		[CVOL=device=serial]
		[EXPAND=n]
		[UNCATLG]
		[CATLG]
		[RENAME=name]

PDS=name
specifies the fully qualified name of the partitioned data set to be copied.

TO=device=serial
specifies the device type and volume serial number of the volume on which the partitioned data set is to be copied.

FROM=device=serial
specifies the device type and volume serial number of the volume on which the partitioned data set currently resides, if it is not cataloged.

If the data set is cataloged, FROM should not be written.

CVOL=device=serial
specifies the device type and volume serial number of the control volume at which the catalog search for the partitioned data set is to begin.

If neither FROM nor CVOL is written, the partitioned data set is assumed to be cataloged on the system residence volume.

EXPAND=n
specifies the number of 256-byte records to be added to the directory of the specified partitioned data set.

UNCATLG
specifies that the catalog entry pertaining to the source partitioned data set is to be removed. This parameter should be used only if the source partitioned data set is cataloged.

UNCATLG is ignored if the volume was identified by FROM.

CATLG
specifies that the copied partitioned data set is to be cataloged on the receiving volume, if it is a direct-access volume. If a catalog does not exist on the receiving volume, a new catalog is created.

RENAME=name
specifies that the data set is to be renamed, and indicates the new name.

Moving a Catalog (MOVE CATALOG)

The MOVE CATALOG statement is used to move the entries in a catalog without moving the data sets associated with those entries. Certain entries can be excluded from a move operation by means of the EXCLUDE statement. If the receiving volume contains a catalog, the source catalog is merged with it.

Name	Operation	Operand
[name]	MOVE	CATALOG[=name] TO=device=serial [CVOL=device=serial] [FROM=device=serial]

CATALOG=name
specifies a qualified name. All catalog entries whose names are qualified by this name are moved. If the name is a fully qualified data set name, only the catalog entry that corresponds to that data set is moved.

CATALOG
specifies that all entries in the catalog are to be moved.

TO=device=serial
specifies the device type and volume serial number of the direct-access volume to which the specified catalog entries are to be moved.

CVOL=device=serial
specifies the device type and volume serial number of the control volume at which the search for the catalog is to begin.

FROM=device=serial
specifies the device type and volume serial number of the volume on which an unloaded version of the catalog resides.

If neither FROM nor CVOL is written, the catalog is assumed to reside on the system residence volume.

Copying a Catalog (COPY CATALOG)

The COPY CATALOG statement is used to copy the entries in a catalog without copying the data sets associated with these entries. Certain entries can be excluded from a copy operation with the EXCLUDE statement. If the receiving volume contains a catalog, the source catalog is merged with it.

Name	Operation	Operand
[name]	COPY	CATALOG[=name] TO=device=serial [CVOL=device=serial] [FROM=device=serial]

CATALOG=name
specifies a qualified name. All catalog entries whose names are qualified by this name are copied. If the name is a fully qualified data set name, only the catalog entry that corresponds to that data set is copied.

CATALOG
specifies that all entries in the catalog are to be copied.

TO=device=serial
specifies the device type and volume serial number of the direct-access volume on to which the specified catalog entries are to be copied.

CVOL=device=serial
specifies the device type and volume serial number of the control volume at which the search for the catalog is to begin.

FROM=device=serial
specifies the device type and volume serial number of the volume on which an unloaded version of the catalog resides.

If neither FROM nor CVOL is written, the catalog is assumed to reside on the system residence volume.

Including Additional Data in Move and Copy Operations (INCLUDE)

The INCLUDE statement is used to enlarge the scope of certain moving and copying operations by including a portion of data not explicitly defined in a MOVE/COPY DSGROUP or MOVE/COPY PDS statement. The INCLUDE statement must appear after the MOVE or COPY statement whose function it modifies. Any number of INCLUDE statements can modify a MOVE or COPY statement.

Name	Operation	Operand
[name]	INCLUDE	DSNAME=name [MEMBER=membername] [FROM=device=list CVOL=device=serial]

DSNAME=name
specifies the fully qualified name of a data set.

With MOVE/COPY DSGROUP: the named data set is included in the group.

With MOVE/COPY PDS: a member of the named partitioned data set is included in the operation.

MEMBER=membername

With MOVE/COPY PDS: specifies the name of a member of the partitioned data set named in the DSNAME parameter. This member will be merged with the partitioned data set that was moved or copied. Its record characteristics must be the same as those of the data set with which it is being merged.

This parameter must be included when modifying a MOVE/COPY PDS statement.

FROM=device=list
specifies the volume or volumes on which the data set resides, if the data set is not cataloged.

If the data set is cataloged, FROM should not be specified.

CVOL=device=serial
specifies the device type and volume serial number of the control volume at which the catalog search for the data set is to begin.

If both FROM and CVOL are omitted, the specified data set is assumed to be cataloged on the system residence volume.

Excluding Data from Move and Copy Operations (EXCLUDE)

The EXCLUDE statement is used to restrict the scope of certain moving or copying operations by excluding a specific portion of the data defined in a MOVE/COPY DSGROUP, PDS, or CATALOG statement.

Partitioned data set members excluded from a MOVE PDS operation cannot be recovered (the source data set is scratched). Any number of EXCLUDE statements can modify a MOVE/COPY PDS statement.

Source data sets or catalog entries excluded from a MOVE DSGROUP or MOVE CATALOG operations remain available. Only one EXCLUDE statement can modify a MOVE/COPY DSGROUP or MOVE/COPY CATALOG statement.

Name	Operation	Operand
[name]	EXCLUDE	{DSGROUP=name MEMBER=membername}

DSGROUP=name
specifies a qualified name.

With MOVE/COPY DSGROUP: all data sets whose names are qualified by this name are excluded from the move or copy operation.

With MOVE/COPY CATALOG: all catalog entries whose names are qualified by this name are excluded from the move or copy operation.

MEMBER=membername

With MOVE/COPY PDS: identifies a member to be excluded from the partitioned data set being moved or copied.

Selecting Partitioned Data Set Members (SELECT)

The SELECT statement, used with a MOVE/COPY PDS statement, is used to select members to be moved or copied, and to optionally rename these members. The SELECT and EXCLUDE statements are mutually exclusive when used with a MOVE/COPY PDS statement.

Name	Operation	Operand
[name]	SELECT	MEMBER=(namelist)

MEMBER=(name[,name]...)
identifies the members to be moved or copied. These members belong to the partitioned data set identified in the preceding MOVE/COPY PDS statement.

MEMBER=((name,newname)[,(name,newname)]...)
identifies the members to be moved or copied and gives the new name for each member.

Replacing Partitioned Data Set Members (REPLACE)

The REPLACE statement, used with a MOVE/COPY PDS statement, is used to combine the exclude and include functions for a specified member of a partitioned data set. The replacing function excludes a member from a moving or copying operation and replaces it by including a member from another partitioned data set. The "new" member must have the same name as the "old" member and must possess identical record characteristics. Any number of REPLACE statements can modify a MOVE/COPY PDS statement.

Name	Operation	Operand
[name]	REPLACE	DSNAME=name
		MEMBER=membername
		[FROM=device=serial]
		[CVOL=device=serial]

DSNAME=name
specifies the fully qualified name of the partitioned data set that contains the replacing member.

MEMBER=membername
specifies the name of the member.

FROM=device=serial
specifies the device type and volume serial number of the volume that contains the partitioned data set named in the DSNAME parameter.

If the partitioned data set is cataloged, FROM should not be specified.

CVOL=device=serial
specifies the device type and volume serial number of the control volume at which the catalog search for the partitioned data set containing the replacing member is to begin.

If neither FROM nor CVOL are specified, the partitioned data set is assumed to be cataloged on the system residence volume.

Moving a Volume of Data (MOVE VOLUME)

The MOVE VOLUME statement is used to move all the data sets residing on a specified direct-access volume. Catalog entries associated with the data sets on the source volume are not affected.

Name	Operation	Operand
[name]	MOVE	VOLUME=device=serial TO=device=list [PASSWORD]

VOLUME=device=serial
specifies the device type and volume serial number of the source volume.

TO=device=list
specifies the volume or volumes to which the data sets are to be moved.

PASSWORD
specifies that password-protected data sets contained on the source volume are to be considered eligible for the moving operation.

If PASSWORD is omitted, only data sets that are not password-protected are moved.

Copying a Volume of Data (COPY VOLUME)

The COPY VOLUME statement is used to copy all the data sets residing on a specified direct-access volume. Catalog entries associated with the data sets on the source volume are not affected. The copied data sets will be cataloged on their receiving volumes if CATLG is specified.

Name	Operation	Operand
[name]	COPY	VOLUME=device=serial TO=device=list [PASSWORD] [CATLG]

VOLUME=device=serial
specifies the device type and volume serial number of the source volume.

TO=device=list
specifies the volume or volumes onto which the data sets are to be copied.

PASSWORD
specifies that password-protected data sets contained on the source volume are to be considered eligible for the copying operation.

If PASSWORD is omitted, only data sets that are not password-protected are copied.

CATLG
specifies that copied data sets are to be cataloged on their respective receiving volumes, if they are direct-access volumes. If catalogs do not exist on the receiving volumes, they are created.

Example

Figure 4 illustrates some typical applications of the IEHMOVE utility program.

GROUP1 moves a partitioned data set named P.D.S1 from the volume where it resides to disk storage volume P1. During the moving operation member M5 is excluded, member M2 from data set P.D.S2 is added, and member M4 is replaced by a member with the same name from a third partitioned data set (P.D.S3).

GROUP2 copies a catalog from the system residence volume to disk storage volume P2. All entries beginning with X.Y are deleted from the operation.

GROUP3 moves all cataloged data sets whose names begin with X.Y to volume P1.

GROUP4 moves data set X.Y.Z from disk storage volume P1 to tape volume T3 as the second data set on the volume.

Sample Coding Form

```
//JOB2      JOB   09-550,BROWN
//          EXEC  PGM=IEHMOVE
//SYSPRINT  DD    SYSOUT=A
//DD1       DD    VOLUME=SER=1234,DISP=OLD,UNIT=2302
//DD2       DD    VOLUME=PRIVATE,UNIT=(2311,,DEFER),DISP=(NEW,KEEP)
//SYSUT1    DD    UNIT=2311,VOLUME=SER=123,DISP=OLD
//DD3       DD    VOLUME=PRIVATE,UNIT=(2400-2,,DEFER),DISP=(NEW,KEEP)
//SYSIN     DD    *
GROUP1      MOVE      PDS=P.D.S1,TO=2311=P1
          EXCLUDE     MEMBER=M5
          INCLUDE     DSNAME=P.D.S2,MEMBER=N2
          REPLACE     DSNAME=P.D.S3,MEMBER=M4
GROUP2      COPY      CATALOG,TO=2311=P2
          EXCLUDE     DSGROUP=X.Y
GROUP3      MOVE      DSGROUP=X.Y,TO=2311=P1
GROUP4      MOVE      DSNAME=X.Y.Z,FROM=2311=P1,TO=2400-2=(T3,2)
/*
```

Figure 4. Moving and Copying Data

LISTING SYSTEM CONTROL DATA

The IEHLIST utility program lists system control data. It is executed or invoked with the symbolic name IEHLIST. The utility statements can request the following types of listings:

- Entries in a catalog.
- Entries in the directory of one or more partitioned data sets.
- Entries in a volume table of contents.

Any number of listings can be requested in a single job step.

Listing a Catalog (LISTCTLG)

The LISTCTLG statement is used to request a listing of either the entire catalog or a specified portion of a catalog. The listing includes the fully qualified name of each cataloged data set and the serial number of the volume on which it resides.

Name	Operation	Operand
[name]	LISTCTLG	[VOL=device=serial] [NODE=name]

VOL=device=serial
specifies the device type and volume serial number of the control volume on which the specified portion of the catalog resides.

If CVOL is omitted, the catalog is assumed to reside on the system residence volume.

NODE=name
specifies a qualified name. All data set entries whose names are qualified by this name are listed.

If NODE is omitted, all data set entries are listed.

Listing Partitioned Data Set Directories (LISTPDS)

The LISTPDS statement is used to request a listing of the directories of one or more partitioned data sets that reside on the same volume.

Name	Operation	Operand
[name]	LISTPDS	[VOL=device=serial] DSNAME=(*namelist*)

VOL=device=serial
specifies the device type and volume serial number of the volume on which the partitioned data sets reside.

If VOL is omitted, the data sets are assumed to reside on the system residence volume.

DSNAME=(name[,name]...)
specifies the fully qualified names of the partitioned data sets whose directories are to be listed. A maximum of ten names is allowed. If the list consists of a single name, the parentheses can be deleted.

Listing a Volume Table of Contents (LISTVTOC)

The LISTVTOC statement is used to request a partial or complete listing of the entries in a specified volume table of contents.

Name	Operation	Operand
[name]	LISTVTOC	{DUMP | FORMAT} [DATE=dddyy] [VOL=device=serial] [DSNAME=(namelist)]

DUMP
specifies that the listing is to be in unedited, hexadecimal form.

FORMAT
specifies that the listing is to be edited for each data set into the following categories:

- Data set name.
- Creation date.
- Expiration date.
- Password indication.
- Organization.
- Extent(s).
- Volume serial number.

The last line in the listing indicates how much space remains in the volume table of contents.

DATE=dddyy
specifies that each entry that expires before this date is to be flagged with an asterisk (*). The date is represented by:

ddd day of the year
yy last two digits of the year

If DATE is omitted, no asterisks appear in the listing.

VOL=device=serial
specifies the device type and volume serial number of the volume whose table of contents is to be listed.

If VOL is omitted, the system residence volume is assumed.

DSNAME=(name[,name]...)
specifies the fully qualified names of the data sets whose entries are to be listed. A maximum of ten names is allowed.

If DSNAME is omitted, the entire volume table of contents is listed.

Example

Figure 5 illustrates the three listing functions of the IEHLIST utility program. The operations are described as comments.

Sample Coding Form

```
//JOB3     JOB   09-550,BROWN                          COMMENTS
//        EXEC   PGM=IEHLIST
//SYSPRINT DD    SYSOUT=A
//DD1      DD    VOLUME=SER=1234,DISP=OLD,UNIT=2302
//DD2      DD    VOLUME=PRIVATE,UNIT=(2311,,DEFER),DISP=(NEW,KEEP)
//SYSIN    DD    *
        LISTCTLG    MODE=A.B                          LISTS A.B ENTRIES
        LISTVTOC    DATE=06965,VOL=2311=P2            LISTS VTOC OF P2
        LISTPDS     DSNAME=(P.D.S1,P.D.S2,P.D.S4)     LISTS DIRECTORIES
/*
```

Figure 5. Listing System Control Data

Data set utility programs manipulate data ranging from fields within a logical record to entire data sets. They include:

- IEBCOPY -- a program that copies or merges partitioned data sets.
- IEBGENER -- a program that copies records from a sequential data set or converts a data set from sequential to partitioned organization.
- IEBCOMPR -- a program that compares records in sequential or partitioned data sets.
- IEBPTPCH -- a program that prints or punches records that reside in a sequential or partitioned data set.
- IEBUPDAT -- a program that updates a symbolic library.

Concatenated partitioned data sets or concatenated data sets with unlike attributes (record characteristics or devices) are not supported by these programs.

Job Control Statement Requirements

Data set utility programs can be introduced as jobs or job steps and are executed in response to job control statements and utility statements. The job control statements used are:

- A JOB statement.
- An EXEC statement that identifies the data set utility program to be executed.
- A DD statement named SYSPRINT that defines a sequential data set used for message output.
- DD statements named SYSUT1 *and* SYSUT2 that define the object data sets. Each statement must identify an object data set and its location with DSNAME, UNIT, and VOLUME parameters, and must give its disposition with DISP. The DCB subparameters RECFM, BLKSIZE, and LRECL (if required) must be available to the utility program through either the DD statement or the data set label. Other DD statement parameters can be used as required.
- A DD statement named SYSIN that defines a sequential data set used for utility statement input.

Figure 6 illustrates a set of job control statements used to execute a data set utility program. Execution can be simplified by cataloging procedures, as described in the introduction to "Section 1: System Utilities."

Data set utility programs can also be employed as subroutines by use of the linkage conventions of the IBM System/360 Operating System. At the completion or termination of the utility program, the highest return code encountered within either the program or its exit routines is passed to the calling program. Invocation of utility programs, and linkage conventions, are discussed in Appendix B.

```
//*
//SYSIN    (DD statement for the data set containing utility statements)
//SYSUT2   DD  DSNAME=name,VOLUME=SER=serno,UNIT=name,DISP=(,KEEP),      C
//SYSUT1   DD  DSNAME=name,VOLUME=SER=serno,UNIT=name,DISP=(OLD,KEEP),   C
//SYSPRINT DD  SYSOUT=A
//stepname EXEC PGM=progname
//jobname  (JOB statement)
```

Figure 6. Executing a Data Set Utility Program

COPYING AND MERGING PARTITIONED DATA SET MEMBERS

The IEBCOPY utility program can copy a partitioned data set or, in effect, merge two partitioned data sets. It is executed or invoked with the symbolic name IEBCOPY.

A partitioned data set is expanded or compressed by adding or omitting members from an operation. Merging is accomplished by the copying of members from the input data set to an existing output data set. In addition to being copied from one volume to another, a partitioned data set can be copied onto its own volume, provided its data set name is changed on the output DD statement. The input and output data sets are defined in DD statements named SYSUT1 and SYSUT2, respectively.

Reblocking is requested by specifying a new maximum block length on the DD statement named SYSUT2. Both fixed-length and variable-length records can be reblocked. If keys are present, or if note lists are present and a total copy is requested, reblocking is not permitted. If note lists are present and a selective copy is requested, reblocking is permitted and the note lists are deleted.

Utility Control Statements: The utility control statements for the IEBCOPY program are:

- COPY (optional).
- MEMBER (optional).

The COPY statement, if present, must be the first utility statement. If no utility statements appear in the SYSIN data set, the entire partitioned data set defined by SYSUT1 is copied.

COPY Statement

The COPY statement is required if the copying or merging operation deals only with selected members of a partitioned data set. It is used in conjunction with the MEMBER statement to select members. A copy operation can be either inclusive or exclusive. An inclusive operation is useful for copying a small number of members; an exclusive operation is useful for copying many members. If the COPY statement contains a blank operand, the entire partitioned data set is copied.

Name	Operation	Operand
[name]	COPY	[TYPCOPY=I,MAXNAME=n TYPCOPY=E,MAXNAME=n]

TYPCOPY=I (inclusive operation)
specifies that the members identified in the MEMBER statement are to be included in the operation.

TYPCOPY=E (exclusive operation)
specifies that the members identified in the MEMBER statement are to be excluded from the operation.

MAXNAME=n
specifies a number no less than the total number of member names and aliases listed in the MEMBER statement.

MEMBER Statement

The MEMBER statement is used to identify the one or more members of a partitioned data set that are to be included or excluded from an operation, and to transfer aliases to the output directory. Table 3 illustrates the combined use of COPY and MEMBER statements to direct the functions of the IEBCOPY program.

Name	Operation	Operand
[name]	MEMBER	NAME=(namelist)

NAME=(list of names and aliases)
identifies the members to be included in, or excluded from, the operation. A member can be referred to by its name, its aliases, or both.

In addition to a member name, one or more alias names can be included in the output directory, if they appeared in the input directory. If a member is identified only by an alias, that alias is designated as the member name in the output directory. If a member is identified by more than one alias, the alias with the lowest collating value is designated the member name in the output directory; all other alias names are retained.

Table 3. Use of COPY and MEMBER Statements in the IEBCOPY Program

To Perform a	The SYSIN Data Set Must Contain
Copy of the entire partitioned data set	no control statements OR name COPY
Selective copy (inclusive)	name MEMBER NAME=(list of members to be copied) name COPY TYPCOPY=I,MAXNAME=n
Selective copy (exclusive)	name MEMBER NAME=(list of members to be excluded) name COPY TYPCOPY=E,MAXNAME=n

Example

Figure 7 illustrates the selective copying of members named MBR3, MBR6, and MBR9 from the partitioned data set identified in SYSUT1.

Sample Coding Form

```
//COPY     JOB   09-660,D.P.BROWN
//        EXEC  PGM=IEBCOPY
//SYSPRINT DD    SYSOUT=A
//SYSUT1   DD    (Parameters defining the input data set.)
//SYSUT2   DD    (Parameters defining the output data set.)
//SYSIN    DD    *
           COPY  TYPCOPY=I,MAXNAME=3
           MEMBER NAME=(MBR3,MBR6,MBR9)
/*
```

Figure 7. Copying Partitioned Data Set Members

COPYING AND MODIFYING RECORDS

The IEBGENER program can copy a sequential data set or generate a partitioned data set from sequential input. It is executed or invoked with the symbolic name IEBGENER. Both operations can include the following editing facilities:

- Rearrangement and omission of data fields.
- Addition of literal fields.
- Conversion of data fields.

In addition, user exits are provided at appropriate places for routines that process user labels, manipulate input data, create keys, and handle uncorrectable input/output errors. Interpretation of exit routine return codes by this program is discussed in Appendix A.

The input data set, defined in a DD statement named SYSUT1, must be a sequential data set, and can contain fixed-length, variable-length, or undefined-length records.

The output data set, defined in a DD statement named SYSUT2, can be either sequential or partitioned. Fixed-length and variable-length records can be reblocked by the specification of a new maximum block length on the output DD statement. If the output data set resides on a direct-access device, keys cannot be retained with reblocking, except through the use of the appropriate user exit.

Utility Control Statements: The utility control statements for the IEBGENER program are:

- GENERATE (optional).
- EXITS (optional).
- MEMBER (optional).
- RECORD (optional).

The GENERATE statement, if present, must be the first utility statement. If user exits are used, the EXITS statement follows the GENERATE statement. If no utility statements appear in the SYSIN data set, the entire SYSUT1 data set is copied sequentially.

GENERATE Statement

The GENERATE statement must appear before other statements. If it contains errors, or is inconsistent with other statements, the program is terminated.

Name	Operation	Operand
[name]	GENERATE	[MAXNAME=n]
		[MAXFLDS=n]
		[MAXGPS=n]
		[MAXLITS=n]

MAXNAME=n
: specifies a number that is no less than the total number of member names and aliases appearing in subsequent MEMBER statements. MAXNAME is required if there are one or more MEMBER statements.

MAXFLDS=n
: specifies a number that is no less than the total number of FIELD parameters appearing in subsequent RECORD statements. MAXFLDS is required if there are any FIELD parameters in subsequent RECORD statements.

MAXGPS=n
: specifies a number that is no less than the total number of IDENT parameters appearing in subsequent RECORD statements. MAXGPS is required if there are any IDENT parameters in subsequent RECORD statements.

MAXLITS=n
: specifies a number that is no less than the total number of characters contained in the FIELD literals of subsequent RECORD statements. MAXLITS is required if the FIELD parameters of subsequent RECORD statements contain literals.

EXITS Statement

The EXITS statement is used to identify exit routines supplied by the user. The exits OUTHDR and OUTTLR are ignored if the output data set is partitioned. Linkages to and from exit routines are discussed in Appendix A.

Name	Operation	Operand
[name]	EXITS	[INHDR=routinename] [OUTHDR=routinename] [INTLR=routinename] [OUTTLR=routinename] [KEY=routinename] [DATA=routinename] [IOERROR=routinename]

INHDR=routinename
specifies the symbolic name of a routine that processes user input header labels.

OUTHDR=routinename
specifies the symbolic name of a routine that creates user output header labels.

INTLR=routinename
specifies the symbolic name of a routine that processes user input trailer labels.

OUTTLR=routinename
specifies the symbolic name of a routine that creates user output trailer labels.

KEY=routinename
specifies the symbolic name of a routine that creates the output record key.

DATA=routinename
specifies the symbolic name of a routine that modifies the input record before it is processed.

IOERROR=routinename
specifies the symbolic name of a routine that handles permanent input/output error conditions.

MEMBER Statement

The MEMBER statement provides the name and aliases of a member that is to be created. The record statements that follow a MEMBER statement pertain to the named member until a new MEMBER statement is encountered. If no MEMBER statements are included, the output data set is organized sequentially.

Name	Operation	Operand
[name]	MEMBER	NAME=(name[,alias]...)

NAME=(name[,alias]...)
specifies a member name followed by a list of its aliases.

RECORD Statement

The RECORD statement is used to define a "record group", and to supply editing information. A record group consists of one or more records that are to be edited identically. The RECORD statement defines a record group by identifying the last record of the group with a literal name.

The RECORD statement can contain any number of FIELD parameters. If no RECORD statement is used, the entire input data set or member is copied without editing.

Name	Operation	Operand
[name]	RECORD	[IDENT=(length,'name',input-location)] [FIELD=([length],[input location-or-'literal'],[conversion],[output-location])...]

IDENT=(length,'name',input-location)
identifies the last record of the input record group to which the FIELD parameters apply.

length: specifies the length (in bytes) of the identifying name. The length cannot exceed 8 characters.

'name': specifies the exact literal that identifies the last input record of a record group.

input-location: specifies the starting location of the field that contains the identifying name in the input records.

If IDENT is omitted or if no match for 'name' is found, the remainder of the input data is considered to be in one record group, i.e., subsequent RECORD and MEMBER statements are ignored.

FIELD=([length],[input-loc'n-or-'literal'],[conversion],[output-location])...
specifies the field processing and editing information. FIELD parameters can determine the logical record length of output records.

length: specifies the length (in bytes) of the input field to be processed. If a length is not specified, 80 is assumed.

input-location or 'literal': specifies the starting byte of the field to be processed or, if enclosed in apostrophes, the literal (maximum length of 40 bytes) to be placed in the specified output location. If an input location is not specified, 1 is assumed. If the literal contains apostrophes, each must be written as two consecutive apostrophes.

conversion: specifies a 2-byte code that indicates the type of conversion to be performed on this field, as follows:

Code	Conversion	Output length (Input length is L)
PZ	Packed to unpacked decimal mode	2L-1
ZP	Unpacked to packed decimal mode	L/2+C *
HE	H-set BCD to EBCDIC mode	L

*If L is odd, C is 1/2; if L is even, C is 1.

If a conversion is not specified, the field will be moved to the output area without change.

output-location: specifies the starting location of this field in the output records. If an output location is not specified, 1 is assumed.

Contents of unspecified fields in the output records are not erased or modified by the IEBGENER program. These contents remain the same as they were before the program was executed.

Examples

Figure 8 illustrates the generation of a two-member partitioned data set from a sequential data set. The last record of GROUP1 is identified with the literal 'REC15'; the identification field is located in bytes 2 through 6 of the input records. GROUP2 contains the remainder of the input data set.

Figure 9 illustrates the editing of records in a sequential data set. The records are edited in the following manner:

- Asterisks are placed in positions 1 to 10.
- Bytes 1 to 5 of the input record are converted from H-set BCD to EBCDIC mode and moved to positions 11 to 15.
- An equal sign is placed in byte 16.
- Bytes 20 to 35 of the input record are converted from packed to unpacked decimal mode and moved to position 17 to 47.
- Blanks are placed in positions 48 to 60.

Two exit routines, ERR and INHDRTN, are used.

```
                                Sample Coding Form
//GEN      JOB   09-660,D.P.BROWN
//         EXEC  PGM=IEBGENER
//SYSPRINT DD    SYSOUT=A
//SYSUT1   DD    (Parameters defining the input sequential data set.)
//SYSUT2   DD    (Parameters defining the output partitioned data set.)
//SYSIN    DD    *
         GENERATE MAXNAME=5,MAXFLDS=2,MAXGPS=2
         EXITS    INHDR=INHDRTN,INTLR=INTLRTN,KEY=KEYEXIT
         MEMBER   NAME=(MBR3,ALS1,ALS2)
GROUP1   RECORD   IDENT=(5,'REC15',2),FIELD=(,,HE)
         MEMBER   NAME=(MBR4,ALS3)
GROUP2   RECORD   FIELD=(,,HE)
/*
```

Figure 8. Generating a Partitioned Data Set From Sequential Input

```
                                Sample Coding Form
//EDIT     JOB   09-660,D.P.BROWN
//         EXEC  PGM=IEBGENER
//SYSPRINT DD    SYSOUT=A
//SYSUT1   DD    (Parameters defining the input sequential data set.)
//SYSUT2   DD    (Parameters defining the output sequential data set.)
//SYSIN    DD    *
         GENERATE MAXFLDS=5,MAXLITS=24
         EXITS    IOERROR=ERR,INHDR=INDRTN
         RECORD   FIELD=(10,'**********',,1),FIELD=(5,1,HE,11),          C
               FIELD=(1,'=',,16),FIELD=(16,20,PZ,17),                    C
               FIELD=(13,'             ',,48)
/*
```

Figure 9. Editing the Records of a Sequential Data Set

COMPARING RECORDS

The IEBCOMPR utility program compares two identically organized data sets at the logical record level. The data sets can be either sequential or partitioned. In addition, user exits are provided at appropriate places for routines that process user labels, handle error conditions, and modify source records. Interpretation of exit routine return codes by this program is discussed in Appendix A. This program is executed or invoked with the symbolic name IEBCOMPR. The input data sets to be compared are defined in the DD statements named SYSUT1 and SYSUT2.

Two sequential data sets are considered "unequal" if:

- One data set contains more records than the other.
- Corresponding records are not identical.
- Corresponding keys are not identical.

If comparison of two records shows them to be unequal, both record and block numbers, the names of the DD statements that define the data sets, and the unequal records are printed.

Two partitioned data sets can be compared if all the names in one directory have counterpart entries in the other. The comparison is made on members identified by these entries, and corresponding user data. Two members are considered "unequal" if:

- Corresponding members do not contain an equal number of records.
- Note lists are not in the same position within corresponding members.
- Corresponding records are not identical.
- Corresponding keys are not identical.

If comparison of two records shows them to be unequal, the record and block numbers, the names of the DD statements that define the data sets, and the unequal records are printed.

Utility Control Statements: The utility control statements for the IEBCOMPR program are:

- COMPARE (optional).
- EXITS (optional).

The COMPARE statement, if present, must be the first utility statement. If no utility statements appear in the SYSIN data set, the two input data sets are assumed to be sequential.

COMPARE Statement

The COMPARE statement is used to indicate the type of data set organization.

Name	Operation	Operand
[name]	COMPARE	{TYPORG=PS / TYPORG=PO}

TYPORG=PS
specifies that the input data sets are organized sequentially.

TYPORG=PO
specifies that the input data sets are partitioned.

If TYPORG is omitted, the input data sets are assumed to be sequential.

EXITS Statement

The EXITS statement is used to identify exit routines supplied by the user. Exits to label processing routines are ignored if partitioned data sets are being compared. Linkages to and from exit routines are discussed in Appendix A.

Name	Operation	Operand
[name]	EXITS	[INHDR=routinename] [INTLR=routinename] [ERROR=routinename] [PRECOMP=routinename]

INHDR=routinename
specifies the symbolic name of a routine that processes user input header labels.

If INHDR is omitted, the user header labels from both input data sets are printed.

INTLR=routinename
specifies the symbolic name of a routine that processes user input trailer labels.

If INTLR is omitted, the user trailer labels from both input data sets are printed.

ERROR=routinename
specifies the symbolic name of an error routine to be given control after each unequal comparison.

Sequential Data Sets: If ERROR is omitted and ten successive unequal comparisons occur, processing is terminated.

Partitioned Data Sets: If ERROR is omitted and ten successive unequal comparisons occur, processing continues with the next member.

PRECOMP=routinename
specifies the symbolic name of a routine that modifies logical records from either or both of the input data sets before they are compared.

If PRECOMP is omitted, neither input data set is modified.

Example

Figure 10 illustrates the comparing of two sequential data sets. Three exit routines, INHDRTN, INTLRTN, and MODIFY, are used.

Sample Coding Form

```
//COMP      JOB   09-660,D.P.BROWN
//          EXEC  PGM=IEBCOMPR
//SYSPRINT  DD    SYSOUT=A
//SYSUT1    DD    (Parameters defining the first input data set.)
//SYSUT2    DD    (Parameters defining the second input data set.)
//SYSIN     DD    *
          COMPARE TYPORG=PS
          EXITS   INHDR=INHDRTN,INTLR=INTLRTN,PRECOMP=MODIFY
/*
```

Figure 10. Comparing the Records in Two Sequential Data Sets

PRINTING AND PUNCHING RECORDS

The IEBPTPCH utility program prints or punches all, or selected portions of, a sequential or partitioned data set. It is executed or invoked with the symbolic name IEBPTPCH, and has the following optional editing facilities:

- Rearrangement and omission of data fields.
- Conversion of data fields or records from packed to unpacked decimal mode and from alphameric to hexadecimal representation.

In addition, user exits are provided at appropriate places for routines that process user labels and manipulate input data. Interpretation of exit routine return codes by this program is discussed in Appendix A. Directories of partitioned data sets can be printed or punched by being defined as sequential data sets.

The input data set, defined in a DD statement named SYSUT1, can be either sequential or partitioned, with variable-length, fixed-length, or undefined-length records. The input mode may be hexadecimal or EBCDIC, packed or unpacked decimal. Packed decimal mode should be converted to unpacked decimal mode to ensure that all characters are printable.

The output data set, defined in a DD statement named SYSUT2, is printed or punched. Title and subtitle records, if requested, appear at the top of each page of printed output or at the start of the deck of punched output.

Records can be printed or punched to meet either standard specifications or user specifications. The standard specifications are as follows:

- The output mode (hexadecimal or EBCDIC, packed or unpacked decimal) is the same as the input mode.
- Each logical record begins on a new printed line or punched card.
- Each printed line consists of 12 groups of eight characters separated by two blanks. Each punched card contains up to 80 contiguous bytes of information.
- Characters that cannot be printed appear as blanks.

Other formats can be specified by RECORD statements, provided that the resulting records do not exceed 120 characters for printed output, or 80 characters for punched output. If sequence numbers are requested, the record to be punched can contain only 72 characters.

Utility Control Statements: The utility control statements for the IEBPTPCH program are:

- PRINT or PUNCH.
- TITLE (optional).
- EXITS (optional).
- MEMBER (optional).
- RECORD (optional).

The PRINT or PUNCH statement must be the first utility statement.

PRINT or PUNCH Statement

The PRINT or PUNCH statement is used to initiate the utility operation.

Name	Operation	Operand
[name]	{PRINT / PUNCH}	{TYPORG=PS / TYPORG=PO}
		[TOTCONV=XE / TOTCONV=PZ]
		[CNTRL=n]
		[INITPG=n / CDSEQ=n]
		[MAXLINE=n / CDINCR=n]
		[STOPAFT=n]
		[SKIP=n]
		[MAXNAME=n]
		[MAXFLDS=n]
		[MAXGPS=n]
		[MAXLITS=n]

TYPORG=PS
specifies that the input data set is organized sequentially.

TYPORG=PO
specifies that the input data set is partitioned.

If TYPORG is omitted, PS is assumed.

TOTCONV=XE
specifies that data is to be printed or punched in 2-character-per-byte hexadecimal representation, e.g., C3 40 F4 F6.

If TOTCONV=XE is not specified, data will be printed or punched in 1-character-per-byte alphameric representation. The above example would appear as C 46.

TOTCONV=PZ
specifies that data in packed decimal mode is to be converted to unpacked decimal mode.

If TOTCONV=PZ is omitted, data is not converted. TOTCONV is not valid if RECORD statements are present.

CNTRL=n
specifies a control character for the output device. For a printer the number indicates line spacing, as follows:

1 - single spacing
2 - double spacing
3 - triple spacing

For a card punch the number selects the card stacker, as follows:

1 - first stacker
2 - second stacker

If CNTRL is omitted, 1 is assumed.

INITPG=n (PRINT only)
specifies the initial page number. Printed pages are numbered sequentially thereafter.

If INITPG is omitted, 1 is assumed.

CDSEQ=n (PUNCH only)
specifies the initial sequence number of a deck of punched cards. This value can be up to 8 digits and occupies columns 73 to 80. Sequence numbering is reinitialized for each member of a partitioned data set.

If CDSEQ is omitted, the cards are not numbered. If n is omitted, 00000000 is assumed.

MAXLINE=n (PRINT only)
specifies the maximum number of lines to a printed page. Spaces, titles, and subtitles must be included in this number.

If MAXLINE is omitted, 60 is assumed.

CDINCR=n (PUNCH only)
specifies the increment to be used in generating sequence numbers.

If CDINCR is omitted, 10 is assumed.

STOPAFT=n
Sequential data sets: specifies the number of logical input records to be printed or punched.

Partitioned data sets: specifies the number of logical input records to be printed or punched in each member.

If STOPAFT is specified and RECORD statements are present, the operation is terminated when the STOPAFT count is satisfied or at the end of the first record group, whichever occurs first.

If RECORD statements are absent and STOPAFT is omitted, processing will continue until the end of data is reached.

SKIP=n
specifies that every "nth" record is to be printed or punched.

If SKIP is omitted, successive logical records are printed or punched.

MAXNAME=n
specifies a number no less than the total number of subsequent MEMBER statements.

If MAXNAME is omitted and there are one or more MEMBER statements, the PRINT or PUNCH request is terminated.

MAXFLDS=n
specifies a number no less than the total number of FIELD parameters appearing in subsequent RECORD statements.

If MAXFLDS is omitted, 0 is assumed.

MAXGPS=n
specifies a number no less than the total number of IDENT parameters appearing in subsequent RECORD statements.

If MAXGPS is omitted, 0 is assumed.

MAXLITS=n
specifies a number no less than the total number of characters contained in the IDENT literals of subsequent RECORD statements.

If MAXLITS is omitted, 0 is assumed.

TITLE Statement

The TITLE statement is used to request title and subtitle records. A first TITLE statement defines the title and a second defines the subtitle.

Name	Operation	Operand
[name]	TITLE	{ITEM=('title', output-location)}...

{ITEM=('title',output-location)}...
specifies title or subtitle information.

'title': specifies the title (or subtitle) literal, (maximum length of 40 bytes) enclosed in apostrophes. If the literal contains apostrophes, each must be written as two consecutive apostrophes.

output location: specifies the starting position at which the title (or subtitle) is to be placed in the output record. If an output location is not specified, 1 is assumed.

EXITS Statement

The EXITS statement is used to identify exit routines supplied by the user. Exits to label processing routines are ignored if the input data set is partitioned. Linkages to and from exit routines are discussed in Appendix A.

Name	Operation	Operand
[name]	EXITS	[INHDR=routinename] [INTLR=routinename] [INREC=routinename] [OUTREC=routinename]

INHDR=routinename
specifies the symbolic name of a routine that processes user input header labels.

INTLR=routinename
specifies the symbolic name of a routine that processes user input trailer labels.

INREC=routinename
specifies the symbolic name of a routine that manipulates each logical record before it is processed.

OUTREC=routinename
specifies the symbolic name of a routine that manipulates each logical record before it is printed or punched. When standard specifications are used, this exit is not available.

MEMBER Statement

The MEMBER statement is used to identify members to be printed or punched. All RECORD statements that follow a MEMBER statement pertain to the named member until a new MEMBER statement is encountered.

If no MEMBER statement appears, and a partitioned set is being processed, all members of the data set are printed or punched. If one or more MEMBER statements appear, the PRINT or PUNCH statement must specify TYPORG=PO.

Name	Operation	Operand
[name]	MEMBER	{NAME=membername {NAME=alias

{NAME=membername
{NAME=alias
identifies a member by its member name or alias.

RECORD Statement

The RECORD statement is used to define a record group that is to be printed or punched to the user's specifications. A record group consists of one or more records to be edited identically. The RECORD statement can contain one IDENT parameter and any number of FIELD parameters.

A RECORD statement referring to a partitioned data set for which no members have been named need contain only FIELD parameters. These are applied to the records in all members of the data set.

If no RECORD statements appear, the entire data set, or named member, is printed or punched to standard specifications. However, if a RECORD statement is used, all data following the record group it defines must be defined with other RECORD statements.

Name	Operation	Operand
[name]	RECORD	[IDENT=(length,'name',[input-location)] [FIELD=(length,[input-location],[conversion],[output-location])...]

IDENT=(length,'name',input-location)
identifies the last record of the record group to which the FIELD parameters apply.

If IDENT is omitted, record processing halts after the last record.

length: specifies the length (in bytes) of the field that contains the identifying name in the input records.

'name': specifies the exact literal that identifies the last record of a record group. If the literal contains apostrophes, each must be written as two consecutive apostrophes.

input-location: specifies the starting location of the field that contains the identifying name in the input records.

FIELD=(length,[input-location],[conversion],[output-location])...
specifies the field processing and editing information.

length: specifies the length (in bytes) of the input field to be processed.

input-location: specifies the starting byte of the field to be processed. If an input location is not specified, 1 is assumed.

conversion: specifies a 2-byte code that indicates the type of conversion to be performed on this field before it is printed, as follows:

Code	Conversion	Output Length (Input Length is L)
PZ	Packed to unpacked decimal mode	2L-1
XE	From alphameric to hexadecimal representation	2L

If conversion is not specified, the field will be moved to the output area without change.

output-location: specifies the starting location of this field in the output records. If an output location is not specified, 1 is assumed. Unspecified fields in the output records appear as blanks on printed output.

Examples

Figure 11 illustrates the statements used to print 30 logical records from a sequential data set. The printed output will be titled THIS IS A SAMPLE, beginning in position 10. The input data will be printed in hexadecimal representation.

Figure 12 illustrates the statements used to punch a partitioned data set. The output will be titled THIS IS A COMPLETE PDS. The initial card sequence number will be 30 with an increment of five. Two exit routines, INEXIT and OUTEXIT, are used. Bytes 1 to 8 of the input records will appear in columns 10 through 17 of the punched output, and bytes 9 through 38 will be printed in hexadecimal representation and placed in columns 20 through 79.

Sample Coding Form

```
//PRINT    JOB    09-660,D.P.BROWN
//         EXEC   PGM=IEBPTPCH
//SYSPRINT DD     SYSOUT=A
//SYSUT1   DD     (Parameters defining the input data set.)
//SYSUT2   DD     (Parameters defining the printed output data set.)
//SYSIN    DD     *
         PRINT   TYPORG=PS,TOTCONV=XE,STOPAFT=30
         TITLE   ITEM=('THIS IS A SAMPLE',10)
/*
```

Figure 11. Printing Records From a Sequential Data Set

Sample Coding Form

```
//PUNCH    JOB    09-660,D.P.BROWN
//         EXEC   PGM=IEBPTPCH
//SYSPRINT DD     SYSOUT=A
//SYSUT1   DD     (Parameters defining the input data set.)
//SYSUT2   DD     (Parameters defining the punched output data set.)
//SYSIN    DD     *
         PUNCH   TYPORG=PO,CNTRL=2,CDSEQ=30,CDINCR=5,                  C
               MAXGPS=1,MAXFLDS=2
         TITLE   ITEM=('THIS IS A COMPLETE PDS',5)
         EXITS   INREC=INEXIT,OUTREC=OUTEXIT
         RECORD  FIELD=(8,1,,10),FIELD=(30,9,XE,20)
/*
```

Figure 12. Punching a Complete Partitioned Data Set

UPDATING SYMBOLIC LIBRARIES

The IEBUPDAT utility program incorporates both IBM- and user-generated source language modifications into symbolic libraries. (A symbolic library is a partitioned data set containing 80-byte records, such as SYS1.PROCLIB, and SYS1.MACLIB.) It is executed or invoked with the symbolic name IEBUPDAT. The program can:

- Add, copy, and replace members.
- Add, delete, replace, and renumber the records within an existing member.
- Assign sequence numbers to the records of a new member.

The IEBUPDAT program uses two input data sets. A DD statement named SYSUT1 defines an old-master partitioned data set. The DD statement named SYSIN defines a sequential data set containing all of the transactions that are to be applied to the old master.

IEBUPDAT also uses two output data sets. A DD statement named SYSUT2 defines a new-master partitioned data set. The DD statement named SYSPRINT defines a sequential data set that reflects either the latest changes applied to the old master or the entire new master.

The input data sets defined by SYSIN and SYSUT1 and the output data set defined by SYSUT2 can contain either blocked or unblocked records with a logical record length of 80 bytes. The output data set can have a blocking factor different from the input data sets.

Notes: If the DD statements SYSUT1 and SYSUT2 define the same data set, the user can make modifications to the old master without creating a new master.

If enough space cannot be allocated for reblocked output records, the update request will be terminated.

EXEC Statement Control Information

The IEBUPDAT program obtains control information through the EXEC statement and the SYSIN data set. The EXEC statement for this program should contain the parameter:

```
PARM=(input,inhdr,intlr)
```

input: specifies either NEW or MOD, as follows:

Input	Meaning
NEW	The input consists of the SYSIN data set.*
MOD	The input consists of both the SYSIN and SYSUT1 data sets.

*Note: The SYSUT1 data set need not be defined if NEW is specified.

If the input is any other value, an error is indicated and the operation terminated. If an input is not specified, MOD is assumed.

inhdr: specifies the symbolic name of a routine that processes the user header label on the SYSIN data set.

intlr: specifies the symbolic name of a routine that processes the user trailer label on the SYSIN data set.

If a value is omitted from the PARM field, its absence must be indicated by a comma (e.g., PARM=(input,,intlr).

All other control information needed to execute the IEBUPDAT program is entered through the SYSIN data set. This information is supplied on a header statement and one or more detail statements and ALIAS statements. An ENDUP statement can be used to indicate the end of the SYSIN input to this program. Figure 13 illustrates the order of the SYSIN control statements.

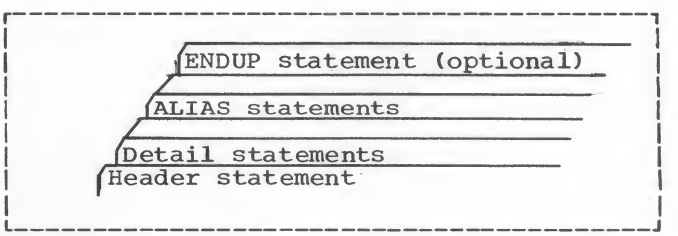

Figure 13. SYSIN Control Statements

The SYSIN data set can contain any number of header statements, each followed by a group of detail statements and ALIAS statements.

Header Statement

A header statement must be provided for each member to be processed. The statements must be in binary collating sequence by member name. The header statement contains:

Columns	Contents
1-2	Period-slash (./)
10	{ ADD / REPL / CHNGE / REPRO }
16	membername,level, source,list,ssi

ADD: indicates that the named member is to be added in its entirety to the new master.

REPL: indicates that the named member is being entered in its entirety as a replacement for a member in the old master.

CHNGE: indicates that modifications are to be made within the named member.

REPRO: indicates that the entire named member is to be copied onto the new master. Members are deleted from a library by being omitted from a series of REPRO operations.

The following parameters, separated by commas, are written beginning in column 16.

membername: specifies the name of the member to which the update transactions are to be applied.

level: specifies the current run number, a 2-digit number from 00 to 99.

source: specifies either 0 or 1, as follows:

0 - indicates user modifications.

1 - indicates IBM modifications.

list: specifies either 0 or 1, as follows:

0 - indicates that the SYSPRINT data set is to contain only modifications and control statements.

1 - indicates that the SYSPRINT data set is to contain the entire updated member and control statements.

ssi: specifies eight bytes of new system status index information that is to be placed in the directory of the new master as the first four hexadecimal bytes of user data. If ssi information is not specified, the user data is copied as it exists in the directory of the old master. System status index information is discussed in detail in the publication IBM System/360 Operating System: Maintenance, Form C27-6918.

Detail Statements

The detail statements contain information to be applied to the member that is named in the header statement. These statements and their formats are discussed in the following paragraphs.

The NUMBR statement is used with CHNGE operations to change the sequence number of one or more logical records within a member, and with ADD and REPL operations to assign sequence numbers to the records within new and replacing members. This statement affects only those sequence numbers that fall in the specified range. A NUMBR statement contains:

Columns	Contents
1-2	A period-slash (./).
10-14	NUMBR
16-23	The sequence number of the first record to be renumbered in CHNGE operations. This field is ignored in ADD and REPL operations.
24	A comma (,).
25-32	The sequence number of the last record to be renumbered in CHNGE operations. This field is ignored in ADD and REPL operations.
33	A comma (,).
34-41	The first new sequence number.
42	A comma (,).
43-50	The increment value of successive new sequence numbers.

Note: All of the sequence numbers must be 8 digit, alphameric fields.

The DELET statement is used to delete one or more logical records within a member. It is used only in conjunction with a CHNGE header statement. A DELET statement contains:

Columns	Contents
1-2	A period-slash (./).
10-14	DELET
16-23	The sequence number of the first logical record to be deleted.
24	A comma (,).
25-32	The sequence number of the last logical record to be deleted.

The logical record statements contain the data to be added to, or used to replace, existing logical records. They are used in conjunction with ADD, REPL, or CHNGE header statements. The logical record statements contain:

Columns	Contents
1-72	The data to be added or used as a replacement.
73-80	The sequence number of the record.

ALIAS Statements

The ALIAS statements create or retain aliases in the new master directory. They can be used in conjunction with any of the header statements. One ALIAS statement must be supplied for each alias. It contains:

Columns	Contents
1-2	A period-slash (./).
10-14	ALIAS
16	An alias name.

Note: If provided, system status index information for the member name is also inserted into the alias entry in the directory of the new master.

ENDUP Statement (optional)

The ENDUP statement can be used to indicate the end of the SYSIN input to this program; it serves as an end-of-data indication if there is no other indication. The ENDUP statement follows the last group of SYSIN control statements and contains:

Columns	Contents
1-2	A period-slash (./).
10-14	ENDUP

Examples

Figure 14 illustrates the cataloging of a set of job control statements in the cataloged procedure library. PROC6 will be added to the cataloged procedure library defined in both SYSUT1 and SYSUT2. The first record of this procedure will contain the sequence number 00000010 with successive records numbered 00000020, 00000030,.... The resulting cataloged procedure can be executed with an EXEC statement specifying PROC=PROC6.

Figure 15 illustrates the replacement of a member in the old master by a new member. In this example, MBR7 (alias ALS7) will replace MBR7 of the library named in SYSUT1. Successive records in the new member will contain the sequence numbers 00000010, 00000020,....

Figure 16 illustrates the deletion of logical records 00000050 through 00000090 from a member of a symbolic library.

Figure 17 illustrates the copying of a member from the old master onto the new master.

Figure 18 illustrates the creation of a three-member library. The members are named LIBMEMB1, LIBMEMB2, and LIBMEMB3.

Sample Coding Form

```
//UPD1     JOB   09-770,D.P.BROWN
//         EXEC  PGM=IEBUPDAT,PARM=MOD
//SYSPRINT DD    SYSOUT=A
//SYSUT1   DD    (Parameters defining the cataloged procedure library.)
//SYSUT2   DD    (Parameters defining the cataloged procedure library.)
//SYSIN    DD    DATA
./       ADD   PROC6,05,0,1
./       NUMBR 00000000,00000000,00000010,00000010
//JOBX   JOB     ---
//STEP1  EXEC    ---
//DD1    DD      ---
         .
         .
//STEP2  EXEC    ---
         .
         .
/*
```

Figure 14. Cataloging Job Control Statements in the Cataloged Procedure Library

Sample Coding Form

```
//UPD2     JOB   09-770,D.P.BROWN
//         EXEC  PGM=IEBUPDAT,PARM=MOD
//SYSPRINT DD    SYSOUT=A
//SYSUT1   DD    (Parameters defining the old master data set.)
//SYSUT2   DD    (Parameters defining the new master data set.)
//SYSIN    DD    *
./       REPL  MBR1,06,0,1
./       NUMBR 00000000,00000000,00000010,00000010
      Logical Record Statements
                .
                .
./       ALIAS ALS7
/*
```

Figure 15. Replacing a Member of a Symbolic Library

```
                                Sample Coding Form
//UPD3      JOB   09-770,D.P.BROWN
//           EXEC  PGM=IEBUPDAT
//SYSPRINT  DD    SYSOUT=A
//SYSUT1    DD    (Parameters defining the old master data set.)
//SYSUT2    DD    (Parameters defining the new master data set.)
//SYSIN     DD    *
./         CHNGE MBRS,13,0,1
./         DELET 00000050,00000090
./         ALIAS ALS5
/*
```

Figure 16. Deleting a Record From a Symbolic Library

```
                                Sample Coding Form
//UPD4      JOB   09-770,D.P.BROWN
//           EXEC  PGM=IEBUPDAT,PARM=MOD
//SYSPRINT  DD    SYSOUT=A
//SYSUT1    DD    (Parameters defining the old master data set.)
//SYSUT2    DD    (Parameters defining the new master data set.)
//SYSIN     DD    *
./         REPRO MBRZ,12,0,1
/*
```

Figure 17. Copying a Member of a Symbolic Library

```
                                Sample Coding Form
//UPD5      JOB   09-770,D.P.BROWN
           EXEC  PGM=IEBUPDAT,PARM=NEW
//SYSPRINT  DD    SYSOUT=A
//SYSUT2    DD    (Parameters for creating a master data set)
//SYSIN     DD    *
./         ADD   LIBMEMB1,01,0,1,1234ABCD
        Logical Record Statements
             .
             .
             .
./         ADD   LIBMEMB2,01,0,1,1234EFGH
        Logical Record Statements
             .
             .
             .
./         ADD   LIBMEMB3,01,0,1,1234JKLM
        Logical Record Statements
             .
             .
             .
/*
```

Figure 18. Creating a Three Member Library

Independent utility programs operate outside, and in support, of the IBM System/360 Operating System. They include:

- DASDI -- a program that initializes and assigns alternate tracks to a direct-access volume.
- DUMP/RESTORE -- a program that dumps and restores the data contents of a direct-access volume.

Utility Control Statement Requirements

Independent utility programs require a JOB statement, a MSG statement, a group of statements that describe the function to be performed, and an END statement.

The JOB statement indicates the beginning of a job.

Name	Operation	Operand
[name]	JOB	[user information]

The MSG statement defines an output device for operator messages. It follows the JOB statement and precedes a group of function-defining statements that are associated with one of the independent utility programs.

Name	Operation	Operand
[name]	MSG	TODEV=xxxx TOADDR=cuu

TODEV=xxxx
specifies the device type of the message output device.

TOADDR=cuu
specifies the channel number (c) and unit number (uu) of the message output device.

The END statement denotes the end of job. It appears after the last function-defining statement.

Name	Operation	Operand
[name]	END	[user information]

Operating Procedure

Independent utility programs are loaded as card decks or as card images on tape. Control statements for the requested program can follow the last card or card image of the program, or can be entered on a separate input device. To execute DASDI or DUMP/RESTORE:

1. Place the object program deck in the reader or mount the tape reel that contains the object program.

2. Load the object program from the reader or tape drive by setting the load selector switches and depressing the console LOAD key. When the program is loaded, the wait state is entered and the console lights display the hexadecimal value FFFF.

3. Define the control statement input device in one of the following two manners:

   a. Depress the REQUEST key of the console typewriter. The message DEFINE INPUT DEVICE is printed. Enter the following message from the console typewriter:

      INPUT=xxxx cuu

      where xxxx is the device type, c is the channel address, and uu is the unit address. The device type can be 1402, 1442, 2400, or 2540.

   b. (Use only if the console typewriter is not available.) Enter one of the following numbers at storage location 0110 (hexadecimal):

      1cuu for a 1442 Card Read Punch.
      2cuu for a 2400 Nine-Track Magnetic Tape Drive.
      0cuu for a 2540 Card Read Punch.

      Depress the console INTERRUPT key and the START key.

4. Control statements are printed on the message output device. At the end of the job, END OF JOB is printed on the message output device and the program enters the wait state.

INITIALIZING AND ASSIGNING ALTERNATE TRACKS ON DIRECT-ACCESS VOLUMES

The DASDI independent utility program performs two separate functions: it initializes direct-access volumes for use with the operation system, and assigns alternate tracks on disk storage volumes. A single job can initialize one volume or assign alternates for specified tracks on one volume. DASDI jobs can be performed continuously by stacking complete sets of control statements. The last job is followed by an END statement and should not be followed by any other control statement.

INITIALIZING A DIRECT-ACCESS VOLUME

The first function of the DASDI program is used to initialize a disk storage volume as follows:

- Writes standard home address and track descriptor records.
- Checks for bad tracks and automatically assigns alternates, if necessary (disk storage volumes, only).
- Writes IPL records on track zero (records 1 and 2).
- Writes initial volume label (record 3) and provides space for additional labels, if requested.
- Writes volume table of contents (VTOC).
- Erases the remainder of each track.
- Writes IPL program, if requested.

The DASDI program requires the following function-defining statements to initialize direct-access volumes:

1. DADEF Statement.
2. VLD Statement.
3. VTOCD Statement.
4. IPLTXT Statement (optional).

These statements must appear in this sequence. Continued control statements are not printed on the message output device.

DADEF Statement

The DADEF statement defines the direct-access volume to be initialized.

Name	Operation	Operand
[name]	DADEF	TODEV=xxxx TOADDR=cuu [IPL=YES] [VOLID=serial VOLID=SCRATCH]

TODEV=xxxx
: specifies the device type of the direct-access device.

TOADDR=cuu
: specifies channel number (c) and unit number (uu) of the device.

IPL=YES
: specifies that an IPL program is to be written on the volume. An IPL initialization program must be written on a device to be used for system residence.

  If IPL is omitted, an IPL program is not written.

VOLID=serial
: specifies the volume serial number of the volume to be initialized.

  If "serial" matches the volume serial number found on the volume to be initialized, the operation proceeds. If it does not match, the operator is notified.

VOLID=SCRATCH
: specifies that no volume serial number check is to be made.

  If VOLID is not specified and the volume to be initialized contains a volume serial number, the operator is notified.

VLD Statement

The VLD statement contains information for constructing an initial volume label and allocating space for additional labels.

Name	Operation	Operand
[name]	VLD	NEWVOLID=serial {VOLPASS= 1} {VOLPASS=0 } [OWNERID=xxxxxxxxxx] [ADDLABEL=n]

NEWVOLID=serial
: specifies a 6-character volume serial number.

VOLPASS=1
specifies that the volume security bit is to be set to 1.

VOLPASS=0
specifies that the volume security bit is to be set to 0.

If VOLPASS is omitted, the volume security bit will be set to 0.

OWNERID=xxxxxxxxxx
specifies a 1- to 10-character field that identifies the owner of the volume.

If OWNERID is omitted, no identification is given.

ADDLABEL=n
specifies a number between one and seven that indicates the total number of additional labels for which space is to be allocated.

If ADDLABEL is omitted, 0 is assumed.

VTOCD Statement

The VTOCD statement contains information for controlling the location of the volume table of contents.

Name	Operation	Operand
[name]	VTOCD	STRTADR=nnnnn EXTENT=nnnn

STRTADR=nnnnn
specifies the 1- to 5-byte track address, relative to the beginning of the volume, at which the volume table of contents is to begin. The VTOC cannot occupy track 00 or any alternate track.

EXTENT=nnnn
specifies the length of the volume table of contents in tracks. The number of entries per track for each type of device is given below.

Device	VTOC Entries/Track
2311	16
2302	22

IPLTXT Statement

The IPLTXT statement separates utility control statements from IPL program text statements. It is required only when IPL text is included. The statement consists of the operation IPLTXT, followed with blanks.

When IPL text is included, the END statement must contain the operation END in columns 2 to 4.

Example

Figure 19 illustrates the initialization of an IBM 2311 disk storage volume for later use as a system residence volume. An IPL program is included as a load module in standard TXT format.

Sample Coding Form

```
INIT      JOB    'INITIALIZE 2311'              COMMENTS
          MSG    TODEV=1403,TOADDR=00E          MESSAGE OUTPUT
          DADEF  TODEV=2311,TOADDR=108,IPL=YES  VOLUME DEFINITION
          VLD    NEWVOLID=P1,OWNERID=BROWN,     VOLUME LABEL        C
                     ADDLABEL=2                   DEFINITION
          VTOCD  STRTADR=1,EXTENT=9             VTOC DEFINITION
          IPLTXT                                DELIMITER
 TXT                                            IPL PROGRAM
  .                                               .
  .          IPL Program                          .
  .                                               .
 TXT
 END
```

Figure 19. Initializing a Direct-Access Volume

ASSIGNING AN ALTERNATE TRACK

The second function of the DASDI program is used specifically to assign alternate tracks on a disk storage volume. Before an alternate is assigned, the track in question is checked. If it is found to be bad, an alternate is assigned and the address made known to the operator.

If the track in question is already assigned as an alternate, a new alternate is assigned. If the track in question is an unassigned alternate, it is flagged to prevent its future use.

GETALT Statement

Any number of alternate tracks on a volume can be assigned in a single job by including one GETALT statement for each track.

```
┌──────┬─────────┬──────────────────────┐
│Name  │Operation│Operand               │
├──────┼─────────┼──────────────────────┤
│[name]│GETALT   │TODEV=xxxx            │
│      │         │TOADDR=cuu            │
│      │         │TRACK=ccccchhhh       │
│      │         │[BYPASS=YES]          │
│      │         │VOLID=serial          │
└──────┴─────────┴──────────────────────┘
```

TODEV=xxxx
: specifies the device type of the direct-access device.

TOADDR=cuu
: specifies the channel number (c) and unit number (uu) of the direct-access device.

TRACK=ccccchhhh
: specifies address of the track for which an alternate is requested, where cccc is the cylinder number and hhhh is the head number.

BYPASS=YES
: specifies that the program's bad-track checking feature is to be bypassed.

  If BYPASS is omitted, the program assigns an alternate only if it finds that the specified track is bad.

VOLID=serial
: specifies the volume serial number of the volume to which an alternate track is to be assigned.

  If "serial" matches the volume serial number found on this volume, the alternate track assignment proceeds. If it does not match, the operator is notified.

Example

Figure 20 illustrates the assignment of three alternate tracks to a disk storage volume, without re-initialization of the volume. The program's bad-track checking feature is bypassed when the first two of the three tracks are assigned.

```
┌──────────────────────────────────────────────────────────────────────────────┐
│                                Sample Coding Form                            │
├──────────────────────────────────────────────────────────────────────────────┤
│ALTRK      JOB   'ASSIGN ALTERNATE TRACKS ON 2311' COMMENTS                   │
│           MSG   TODEV=2400,TOADDR=180             MESSAGE OUTPUT           │
│STMT1    GETALT  TODEV=2311,TOADDR=190,TRACK=006F0001,                  C     │
│               BYPASS=YES,VOLID=P2                                            │
│STMT2    GETALT  TODEV=2311,TOADDR=190,TRACK=00910004,                  C     │
│               BYPASS=YES,VOLID=P2                                            │
│STMT3    GETALT  TODEV=2311,TOADDR=190,TRACK=004B0007,                  C     │
│               VOLID=P2                                                       │
│           END                                                                │
└──────────────────────────────────────────────────────────────────────────────┘
```

Figure 20. Assigning Alternate Tracks on a Disk Storage Volume

DUMPING AND RESTORING A DIRECT-ACCESS VOLUME

The DUMP/RESTORE program dumps and restores the data on direct-access volumes. The data contents of a direct-access volume (all data except the home address) can be "dumped" onto an IBM 2311 disk storage volume or a magnetic tape, and restored onto a direct-access volume that resides on the same type of device as the source volume. Both the source volume and the volume onto which data is to be restored must have been initialized to IBM System/360 Operating System specifications. This utility is useful for preparing transportable copies and backup copies of direct-access volume contents.

DUMP Statement

The DUMP statement is used to identify both the source volume whose contents are to be dumped and the receiving volume. The data contents of the entire source volume is dumped, including any data on alternate tracks. If both the source and receiving volumes reside on 2311 Disk Storage Drives, the data on the receiving volume is an exact replica of the source data and need not be restored.

Name	Operation	Operand
[name]	DUMP	FROMDEV=xxxx FROMADDR=cuu TODEV=xxxx TOADDR=cuu [VOLID=serial-list]

FROMDEV=xxxx
: specifies the device type of the source device.

FROMADDR=cuu
: specifies channel number (c) and unit number (uu) of the source device.

TODEV=xxxx
: specifies the device type of the receiving device.

TOADDR=cuu
: specifies the channel number (c) and unit number (uu) of the receiving device.

VOLID=serial[,serial]...
: specifies the volume serials of the receiving volumes onto which data is to be dumped. (VOLID is required when the receiving volume has been initialized to operating system specifications.)

  If "serial" matches the volume serial number found on the receiving volume, the dump operation proceeds. If it does not match, the operator is notified.

  If VOLID is not specified and the receiving volume contains a volume serial number, the operator is similarly notified.

VDRL Statement

The VDRL (volume dump/restore limits) statement is used to specify the upper and lower limits of a partial dump. If a track within these limits has had an alternate assigned to it, the data on the alternate track is included in the dump. When the VDRL statement is used, it must be preceded by a DUMP statement and must be followed by an END statement.

Name	Operation	Operand
[name]	VDRL	BEGIN=nnnnn [END=nnnnn]

BEGIN=nnnnn
: specifies a 1- to 5-byte relative track address that identifies the first track to be dumped.

END=nnnnn
: specifies the relative track address of the last track to be dumped. If only one track is to be dumped, this address is the same as the beginning address.

  If END is omitted, the last track of the volume, excluding those tracks reserved as alternates, is assumed to be the upper limit.

RESTORE Statement

The RESTORE statement is used to identify both the source volume whose data contents are to be restored and the receiving volume.

Name	Operation	Operand
[name]	RESTORE	FROMDEV=xxxx
		FROMADDR=cuu
		TODEV=xxxx
		TOADDR=cuu
		VOLID=serial

FROMDEV=xxxx
specifies the device type of the source device.

FROMADDR=cuu
specifies the channel number (c) and unit number (uu) of the source device.

TODEV=xxxx
specifies the device type of the receiving device. This device type must be the same as the device containing the volume originally dumped.

TOADDR=cuu
specifies the channel number (c) and unit number (uu) of the receiving device.

VOLID=serial
specifies the volume serial number of the receiving volume.

If "serial" matches the volume serial number found on the receiving volume, the restore operation proceeds. If it does not match, the operator is notified.

Examples

Figure 21 illustrates the dumping of a direct-access volume onto a tape volume. Figure 22 illustrates the restoring of dumped data onto a direct-access volume.

Sample Coding Form

```
DUMP       JOB     DUMP 2311 ONTO TAPE              COMMENTS
           MSG     TODEV=1052,TOADDR=103            MESSAGE OUTPUT
          DUMP     FROMDEV=2311,FROMADDR=203,                        C
               TODEV=2400,TOADDR=120
           END
```

Figure 21. Dumping a Direct-Access Volume

Sample Coding Form

```
RESTORE    JOB    RESTORE 2311 FROM TAPE
           MSG    TODEV=1052,TOADDR=10B
        RESTORE   FROMDEV=2400,FROMADDR=120,TODEV=2311,                C
               TOADDR=202,VOLID=PZ
           END
```

Figure 22. Restoring a Direct-Access Volume

Linking to an Exit Routine

Data set utility programs perform linkage operations by using the LINK macro-instruction. This macro-instruction contains the symbolic name of the entry point to an exit routine and, if required, a list of parameters. Table 4 shows the ordered parameter list for each available exit. (Format and contents of the DCB are presented in the publication IBM System/360 Operating System: Control Program Services, Form C28-6541.)

At the time of the linkage operation:

- General register 1 contains the starting address of the parameter list.
- General register 13 contains the address of the register save area.
- General register 14 contains the address of the return point in the utility program.
- General register 15 contains the address of the entry point to the exit routine.

Registers 1 through 14 must be restored to these values before control is returned to the utility program.

The exit routine must be contained in either the job library or the link library.

Table 4. Parameter Lists for Exit Routines

Program	Exit	Parameters (In Order of Appearance in Parameter List)
COPYPDS	None	
GENERATE	INHDR	Address of input header label; address of DCB.
	OUTHDR	Address at which user output label is to be created; address of DCB.
	INTLR	Address of input trailer label; address of DCB.
	OUTLR	Address at which user output trailer label is to be created; address of DCB.
	KEY	Address at which key is to be placed (record follows key); address of DCB.
	DATA	Address of SYSUT1 record; address of DCB.
	IOERROR	Address of DECB; cause of the error and address of DCB (packed word).[1]
COMPARE	INHDR	Address of input header label; address of DCB.
	INTLR	Address of input trailer label; address of DCB.
	ERROR	Address of DCB for SYSUT1; address of DCB for SYSUT2.[2]
	PRECOMP	Address of SYSUT1 record; length of SYSUT1 record, address of SYSUT2 record; length of SYSUT2 record.
PRINT/PUNCH	INTLR	Address of input header label; address of DCB.
	OUTLR	Address of input trailer label; address of DCB.
	INREC	Address of input record; length of input record.
	OUTREC	Address of output record; length of output record.
UPDATE	inhdr	Address of input header label for SYSIN data set; address of DCB.
	intlr	Address of input trailer label for SYSIN data set; address of DCB.

[1]For the packed-word format of this parameter, refer to the CHECK macro-instruction, discussed in the publication IBM System/360 Operating System: Control Program Services.

[2]The IOBAD pointer in the DCB points to a location that contains the address of the corresponding data event control block (DECB) for these records. The format of the DECB is illustrated as part of the BSAM READ macro-instruction in the publication IBM System/360 Operating System: Control Program Services.

Returning From an Exit Routine

An exit routine returns control to the utility program by means of the RETURN macro-instruction in the exit routine.

Name	Operation	Operand
[name]	RETURN	[(r_1,r_2)] [RC=n] [RC=(15)]

(r_1,r_2)
specifies the range of registers to be reloaded by the utility program from the register save area.

If this parameter is omitted, the registers are considered properly restored by the exit routine.

RC=n
specifies a return code to be placed in the 12 low-order bits of general register 15.

RC=(15)
specifies that general register 15 already contains a valid return code.

If RC is omitted, register 15 is loaded as specified by (r_1,r_2).

The utility programs examine the return code and respond as described in Table 5. Further information on the use of the LINK and RETURN macro-instructions and exit routine linkage is contained in the publication IBM System/360 Operating System: Control Program Services.

Table 5. Action on Return Codes

Type of Exit	Return Code	Action
Label processing exits	0,4	Return code is passed to the OPEN routine.
	16	Utility program is terminated.
	Any other number	Return code is passed to the OPEN routine.
All other exits	0-11 (Set to next lowest multiple of 4, i.e., 0, 4, 8)	Return code is compared to highest previous return code; the higher is saved and the other discarded. At the normal end of job, the highest return code is passed to the calling processor.
	12-16 (Set to either 12 or 16)	Utility program is terminated and this return code is passed to the calling processor.

Utility programs can be invoked by a problem program through the use of the ATTACH or LINK macro-instructions.

The problem program must supply to the utility program:

- The information usually specified in the PARM parameter of the EXEC statement.
- The ddnames of the data sets to be used during processing by the utility program.

Name	Operation	Operand
[name]	{LINK / ATTACH}	EP=progname PARAM=(optionaddr [,ddnameaddr]),VL=1

EP=progname
specifies the symbolic name of the utility program.

PARAM
specifies, as a sublist, address parameters to be passed from the problem program to the utility program. The first full-word in the address parameter list contains the address of information that is usually specified in the PARM parameter of the EXEC statement. The second full-word contains the address of the ddname list.

If standard ddnames are to be used, the second full word can be omitted.

optionaddr
specifies the address of a variable length list containing information usually specified in the PARM parameter of the EXEC statement. This address must be written for all utility programs even though IEBUPDAT is the only program that requires PARM information.

The option list must begin on a halfword boundary (one that is not also a full-word boundary). The two high-order bytes contain a count of the number of bytes in the remainder of the list. (For all programs except IEBUPDAT, the count must be zero.) The option list is free form with each field separated by a comma. No blanks or zeros should appear in the list.

ddnameaddr
specifies the address of a variable length list containing alternate ddnames for the data sets used during utility program processing. If standard ddnames are used, this operand may be omitted.

The ddname list must begin on a halfword boundary (one that is not also a full-word boundary). The two high-order bytes contain a count of the number of bytes in the remainder of the list. Each name of less than eight bytes must be left justified and padded with blanks. If an alternate ddname is omitted from the list, the standard name is assumed. If the name is omitted within the list, the 8-byte entry must contain binary zeros. Names can be omitted from the end by merely shortening the list.

The sequence of the 8-byte entries in the ddname list is as follows:

Entry	Alternate Name
1	00000000
2	00000000
3	00000000
4	00000000
5	SYSIN
6	SYSPRINT
7	00000000
8	SYSUT1
9	SYSUT2

VL=1
specifies that the sign bit of the last full-word of the address parameter list is to be set to 1.

This appendix, message appendixes in other publications, and the publication IBM System/360 Operating System: Control Program Messages and Completion Codes, Form C28-6608, are designed so that the user can select the messages applicable to his installation and incorporate them in a binder.

The IEHLIST Program

IEH101I NO CATALOG ON SPECIFIED VOLUME.

Explanation: No catalog exists on the volume identified in the LISTCTLG statement.

System Action: The request is ignored. (The return code is 8.)

IEH102I THIS VOLUME DOES NOT CONTAIN DATA SET xxx.

Explanation: The data set identified in the LISTVTOC or LISTPDS statement is not contained in the specified volume's table of contents.

System Action: The request is ignored. (The return code is 8.)

IEH103I INVALID CONTROL STATEMENT -- xxx.

Explanation: A utility statement is invalid. (The entire statement is written.)

System Action: The request is ignored. (The return code is 8.)

IEH104I THE PDS ORGANIZATION DOES NOT APPLY FOR DATA SET xxx.

Explanation: The data set identified in the LISTPDS statement is not partitioned.

System Action: The request is ignored. (The return code is 8.)

IEH105I ILLEGAL NODE POINT SPECIFIED, OR INCONSISTENT CATALOG STRUCTURE FOUND -- REQUEST TERMINATED.

Explanation: The node point identified in the LISTCTLG statement is invalid, or an incorrect catalog structure exists.

System Action: The request is ignored. (The return code is 8.)

IEH106I UNAVAILABLE DEVICE TYPE OR VOLUME I.D. SPECIFIED.

Explanation: The VOL parameter of the utility statement is invalid, or the volume cannot be mounted.

System Action: The request is ignored. (The return code is 8.)

IEH107I JOB TERMINATED -- I/O ERROR ON SYSIN.

Explanation: Additional input control statements cannot be read.

System Action: The job is terminated. (The return code is 16.)

IEH108I REQUEST TERMINATED -- I/O ERROR WHILE READING DATA SET.

Explanation: The user's synchronous error exit was taken while a volume table of contents, a catalog, or a partitioned data set was being read.

System Action: The job is terminated. (The return code is 16.)

The IEHPROGM Program

IEH201I INVALID REQUEST ... STATEMENT IGNORED.

Explanation: The utility statement contains an invalid operation.

System Action: The request is ignored. (The return code is 8.)

IEH202I INVALID KEYWORD OR CONTROL STATEMENT ERROR.

Explanation: The utility statement contains an invalid keyword in the operand field, or information that should follow a keyword is absent.

System Action: The request is ignored. (The return code is 8.)

IEH203I CATALOG CANNOT BE LOCATED, OR CONTROL VOLUMES ARE CONNECTED TO EACH OTHER.

Explanation: No catalog exists on the specified control volume, or control volumes are connected incorrectly to each other.

System Action: The request is ignored. (The return code is 8.)

IEH204I STATUS OF THE REQUESTED TASK CANNOT BE DETERMINED. AN UNDEFINED ERROR CODE HAS BEEN ENCOUNTERED.

Explanation: The return code from a system macro-instruction is invalid.

System Action: The request is ignored. (The return code is 8.)

IEH205I INFORMATION IN CONTROL STATEMENT IS (REDUNDANT/NOT SUFFICIENT).

Explanation: Information on the utility statement is either redundant or inadequate.

System Action: The request is ignored. (The return code is 8.)

IEH206I CVOL IS NOT DIRECT-ACCESS.

Explanation: The volume identified by CVOL is not on a direct-access device.

System Action: The request is ignored. (The return code is 8.)

IEH207I STATUS OF USERS REQUEST TO (SCRATCH/RENAME) DATA SET xxx...

```
VOLUME ID    ACTION TAKEN    REASON
   xxx           xxx           xxx
    .             .             .
    .             .             .
    .             .             .
END OF ERROR ANALYSIS LISTING.
```

Explanation: An unusual condition occurred during a SCRATCH or RENAME operation. The message names the data set, identifies the volumes on which the data set resides, states the action taken on each volume, and describes the condition.

System Action: The request is ignored. (The return code is 8.)

IEH208I STATUS OF USERS REQUEST TO SCRATCH THE VOLUME TABLE OF CONTENTS...

```
DATA SET NAME  ACTION TAKEN  REASON
     xxx           xxx         xxx
      .             .           .
      .             .           .
      .             .           .
END OF SCRATCH VTOC.
```

Explanation: An unusual condition occurred during a SCRATCH VTOC operation. The message names each data set, states the action taken, and describes the condition. (The return code is 8.)

IEH209I LIST TRUNCATED TO 1 VOLUME FOR SCRATCH VTOC.

Explanation: More than 1 volume is identified in a SCRATCH VTOC statement.

System Action: Only the first volume in the list is considered. (The return code is 8.)

IEH210I THE MODEL DATA SET CONTROL BLOCK IS NOT AVAILABLE.

Explanation: The required DSCB for a BLDG operation cannot be located on the specified volume.

System Action: The request is ignored. (The return code is 8.)

IEH211I REQUEST CANNOT BE SERVICED ... reason.

Explanation: An unusual condition occurred during a catalog or index operation. The condition is described in detail.

System Action: The request is ignored. (The return code is 8.)

IEH212I REQUIRED VOLUME COULD NOT BE MOUNTED.

Explanation: A device was not allocated for the required volume.

System Action: The request is ignored. (The return code is 8.)

The IEHMOVE Program

IEH301I OPERATION NOT VALID.

Explanation: The control statement operation is invalid.

System Action: The MOVE/COPY request is ignored. (The return code is 8.)

IEH304I INVALID CONTROL STATEMENT SEQUENCE.

Explanation: A MOVE/COPY statement is incorrectly followed by an INCLUDE, EXCLUDE, REPLACE, or SELECT statement.

System Action: The MOVE/COPY request is ignored. (The return code is 8.)

IEH305I ERROR IN OPERAND FIELD.

Explanation: An error exists in the control statement operand field.

System Action: The MOVE/COPY request is ignored. (The return code is 8.)

IEH311I INCOMPLETE REQUEST.

Explanation: The control statement does not contain adequate information to perform the MOVE/COPY operation.

System Action: The MOVE/COPY request is ignored. (The return code is 8.)

IEH313I DATA SET xxx HAS INCORRECT FORMAT FOR UNLOADED DATA SET.

Explanation: The request to move or copy an unloaded data set is ignored because its format is incorrect. The records are apparently out of sequence.

System Action: The MOVE/COPY request is ignored. (The return code is 8.)

IEH319I MEMBER xxx NOT MOVED/COPIED. DUPLICATE NAME IN OUTPUT DATA SET.

Explanation: A member with the same name is contained in the output partitioned data set.

System Action: The request is ignored. (The return code is 8.)

IEH320I MEMBER xxx NOT FOUND IN DATA SET xxx.

Explanation: The named member cannot be located in the partitioned data set.

System Action: The request is ignored. (The return code is 8.)

IEH321I MEMBER xxx NOT MOVED/COPIED. OUTPUT DIRECTORY IS FULL.

Explanation: The directory of the output partitioned data set is full.

System Action: The named member was not moved or copied. (The return code is 8.)

IEH322I I/O ERROR ENCOUNTERED IN MEMBER xxx OF INPUT DATA SET xxx.

Explanation: A permanent error was detected while the named member was being read.

System Action: The MOVE/COPY request is ignored. (The return code is 8.)

IEH323I I/O ERROR ENCOUNTERED IN MEMBER xxx OF OUTPUT DATA SET xxx.

Explanation: A permanent error was detected while the named member was being written.

System Action: The MOVE/COPY request is ignored. (The return code is 8.)

IEH325I INVALID CATLG REQUEST IGNORED.

Explanation: The specified receiving volume is not direct-access.

System Action: The moved or copied data set was not cataloged on this volume as requested. (The return code is 8.)

IEH326I I/O ERROR ENCOUNTERED IN OUTPUT DATA SET xxx.

Explanation: A permanent error was detected while the named data set was being written.

System Action: The request is terminated. The return code is 8.

IEH346I CATALOG CANNOT BE LOCATED, OR CONTROL VOLUMES ARE CONNECTED TO EACH OTHER.

Explanation: No catalog exists on the specified control volume, or control volumes are connected incorrectly to each other.

System Action: The request is ignored. (The return code is 8.)

IEH348I I/O ERROR ENCOUNTERED IN CATALOG.

Explanation: A permanent error was encountered while reading or writing the catalog.

System Action: The request is terminated. (The return code is 8.)

IEH349I UNABLE TO MOUNT VOLUME xxx... action.

Explanation: No device was allocated for the specified volume.

System Action: The request is ignored. (The return code is 8.)

IEH351I DATA SET xxx NOT CATALOGED. SPACE NOT AVAILABLE IN THE CATALOG.

Explanation: The catalog is full.

System Action: The named data set was not cataloged. (The return code is 8.)

IEH354I DATA SET xxx NOT CATALOGED. INDEX STRUCTURE INCONSISTENT.

Explanation: An invalid index structure exists.

System Action: The named data set was not cataloged. (The return code is 8.)

IEH356I DATA SET xxx NOT CATALOGED. INVALID DATA SET NAME.

Explanation: The specified data set name is inappropriate for cataloging.

System Action: The data set was not cataloged. (The return code is 8.)

IEH372I I/O ERROR ENCOUNTERED IN WORK DATA SET.

Explanation: A permanent input/output error was detected while reading or writing the work data set.

System Action: The request is terminated. (The return code is 12.)

IEH373I UNABLE TO MOUNT VOLUME xxx. SOME INCLUDE OR REPLACE REQUESTS IGNORED.

Explanation: The program cannot mount the named volume.

System Action: The INCLUDE or REPLACE requests that refer to the specified volume were ignored. (The return code is 8.)

IEH374I DATA SET xxx NOT FOUND ON VOLUME xxx. INCLUDE OR REPLACE REQUEST IGNORED.

Explanation: The named data set does not reside on the specified volume.

System Action: The INCLUDE or REPLACE statements that refer to the specified data set were ignored. (The return code is 8.)

IEH375I DATA SET xxx IS NOT PARTITIONED. INCLUDE OR REPLACE REQUEST IGNORED.

Explanation: The named data set is not partitioned.

System Action: The INCLUDE request or the including part of the REPLACE request was ignored. (The return code is 8.)

IEH376I RECORD CHARACTERISTICS NOT COMPATIBLE (xxx). INCLUDE OR REPLACE REQUEST IGNORED.

Explanation: An attribute (xxx) of the output data set is not compatible with that of the input data set.

System Action: The INCLUDE request or the including part of the REPLACE request was ignored. (The return code is 8.)

IEH380I MEMBER xxx NOT FOUND IN DATA SET xxx. INCLUDE OR REPLACE REQUEST IGNORED.

Explanation: The named member is not contained in the named partitioned data set.

System Action: The INCLUDE request or the including part of the REPLACE request is ignored. (The return code is 8.)

IEH383I INVALID DEVICE NAME.

Explanation: A device name on the utility statement is invalid.

System Action: The request is ignored. (The return code is 8.)

IEH388I SPACE NOT AVAILABLE FOR WORK DATA SET.

Explanation: Space is not available for the work data set. The MOVE/COPY request was ignored.

System Action: The MOVE/COPY program was terminated. (The return code is 12.)

IEH389I I/O ERROR ENCOUNTERED IN INPUT DATA SET xxx.

Explanation: A permanent error was detected while the input data set was being read.

System Action: The request is terminated. (The return code is 12.)

IEH401I DATA SET xxx NOT MOVED/COPIED ... reason.

Explanation: The named data set was not moved or copied, for the reason given.

System Action: The request is ignored. (The return code is 8.)

IEH402I DATA SET xxx UNLOADED ... reason.

System Action: The data set was unloaded for the reason given. (The return code is 8.)

IEH436I DATA SET xxx, VOLUME xxx, NOT SCRATCHED DUE TO I/O ERROR.

System Action: The named data set was not scratched after the MOVE operation. (The return code is 8.)

This appendix, message appendixes in other publications, and the publication IBM System/360 Operating System: Control Program Messages and Completion Codes are designed so that the user can select the messages applicable to his installation and incorporate them in a binder.

Data set utility messages indicating job termination can be interpreted several ways:

1. If the utility program was invoked, a return code is passed to the calling program with the option to terminate.
2. If the utility program represents one step of a multi-step job, the step is terminated.
3. Otherwise, the job is terminated.

The IEBCOPY Program

IEB101I INVALID OPERATION.

Explanation: The operation field of the above statement is invalid.

System Action: The job is terminated. (The return code is 12.)

IEB102I INVALID KEYWORD.

Explanation: An invalid keyword appears in the operand of the above statement.

System Action: The job is terminated. (The return code is 12.)

IEB103I INVALID PARAMETER VALUE.

Explanation: An invalid parameter value appears in the operand of the above statement.

System Action: The job is terminated. (The return code is 12.)

IEB104I MAXNAME VALUE TOO SMALL.

Explanation: The value in the parameter of the COPY statement is less than the total number of member names and aliases in the MEMBER statement.

System Action: The job is terminated. (The return code is 12.)

IEB105I INVALID REBLOCKING REQUEST.

Explanation: The request for reblocking does not satisfy the necessary conditions.

System Action: No reblocking was performed.

IEB108I INCLUSIVE COPY ASSUMED.

Explanation: The TYPCOPY parameter was not specified in the COPY statement.

System Action: The entire data set is copied. (The return code is 8.)

IEB109I TOTAL COPY ASSUMED.

Explanation: No control statements appeared; the entire input data set is copied.

System Action: The entire data set is copied. (The return code is 8.)

IEB110I xxx NOT FOUND IN DIRECTORY.

Explanation: The named member was not found in the directory of the SYSUT1 data set.

System Action: The specified member is not copied. (The return code is 8.)

IEB118I DIRECTORY READ ERROR.

Explanation: A permanent error was detected while reading the directory of the SYSUT1 data set.

System Action: The job is terminated. (The return code is 12.)

IEB119I DIRECTORY WRITE ERROR.

Explanation: A permanent error was detected while writing the directory of the SYSUT2 data set.

System Action: The job is terminated. (The return code is 12.)

IEB120I PERMANENT INPUT ERROR.

Explanation: A permanent error was detected while reading the SYSUT1 data set.

System Action: The job is terminated. (The return code is 12.)

IEB121I PERMANENT OUTPUT ERROR.

Explanation: A permanent error was detected while writing the SYSUT2 data set.

System Action: The job is terminated. (The return code is 12.)

IEB122I CONTROL STATEMENT INPUT ERROR.

Explanation: A permanent error was detected while reading the SYSIN data set.

System Action: The job is terminated. (The return code is 12.)

IEB123I OUTPUT DIRECTORY FILLED.

Explanation: The directory of the SYSUT2 data set does not contain sufficient space for all the member entries.

System Action: The job is terminated. (The return code is 12.)

IEB124I NO NAMES FOUND IN DIRECTORY.

Explanation: The SYSUT1 data set does not contain identified members.

System Action: The job is terminated. (The return code is 12.)

IEB126I INVALID CONTROL STATEMENT.

Explanation: The construction of the above control statement is invalid.

System Action: The job is terminated. (The return code is 12.)

IEB127I NO MEMBER NAMES FOR PARTIAL COPY.

Explanation: A selective copy is specified and no MEMBER statements are included.

System Action: The job is terminated. (The return code is 12.)

IEB128I DDNAME xxx CANNOT BE OPENED

Explanation: The named DD statement does not exist.

System Action: The job is terminated. (The return code is 12)

User Response: Either correct the ddname if it is misspelled in the DD statement or the ddlist, or insert a new DD statement with this name.

The IEBCOMPR Program

IEB201I INVALID CONTROL STATEMENT.

Explanation: The construction of the above control statement is invalid.

System Action: The job is terminated. (The return code is 12.)

IEB202I INVALID OPERATION.

Explanation: The operation field of the above statement is invalid.

System Action: The job is terminated. (The return code is 12.)

IEB203I INVALID KEYWORD.

Explanation: The operand field of the above statement contains an invalid keyword.

System Action: The job is terminated. (The return code is 12.)

IEB204I INVALID PARAMETER VALUE.

Explanation: An invalid parameter value appears in the above statement.

System Action: The job is terminated. (The return code is 12.)

IEB205I USER DATA FIELDS UNEQUAL.

Explanation: The user data fields of the SYSUT1 and SYSUT2 data sets are unequal.

System Action: The fields are listed and the comparison continues. (The return code is 8.)

IEB210I TRUE NAMES MISSING FROM BOTH SETS.

Explanation: All the names in one directory must have counterpart entries in the other. This condition was not met.

System Action: The job is terminated. (The return code is 12.)

IEB211I KEY LENGTHS ARE NOT EQUAL.

Explanation: The SYSUT1 and SYSUT2 keys are of different lengths.

System Action: The job is terminated. (The return code is 12.)

IEB212I INVALID RECORD FORMAT.

Explanation: The record formats are not standard.

System Action: The job is terminated. (The return code is 12.)

IEB214I FIXED RECORD LENGTHS UNEQUAL.

Explanation: SYSUT1 contains records of a different length than those in SYSUT2.

System Action: The job is terminated. (The return code is 12.)

IEB215I RECORD FORMATS DIFFERENT.

Explanation: The record characteristics of the SYSUT1 and SYSUT2 data sets differ.

System Action: The job is terminated. (The return code is 12.)

IEB217I PERMANENT INPUT ERROR.

Explanation: A permanent error was detected while reading one of the input data sets.

System Action: The job is terminated. (The return code is 12.)

IEB221I RECORDS ARE NOT EQUAL.

Explanation: Two corresponding records do not contain the same data.

System Action: The records are listed and the comparison continues. (The return code is 8.)

IEB222I KEYS ARE NOT EQUAL.

Explanation: Two corresponding keys do not contain the same data.

System Action: The records are listed and the comparison continues. (The return code is 8.)

IEB223I EXTRA RECORD ON SYSUT2.

Explanation: The SYSUT2 data set contains more records than the SYSUT1 data set.

System Action: The records are printed. (The return code is 8.)

IEB224I EXTRA RECORD ON SYSUT1.

Explanation: The SYSUT1 data set contains more records than the SYSUT2 data set.

System Action: The extra records are printed. (The return code is 8.)

IEB225I JOB TERMINATED AFTER EXIT.

Explanation: The return code from an exit routine indicates termination.

System Action: The job is terminated. (The return code is 12 or 16. It is determined by exit routine.)

IEB227I TEN CONSECUTIVE ERRORS.

Explanation: An error routine is not specified and ten successive unequal comparison have occurred.

System Action: If the input data sets are sequential, the job is terminated. If the input data sets are partitioned, processing will continue with the next member. If the current member is the last, the job is terminated. (The return code is 12 for sequential data sets and 8 for partitioned data sets.)

IEB228I DDNAME xxx CANNOT BE OPENED

Explanation: The named DD statement does not exist.

System Action: The job is terminated. (The return code is 12.)

User Response: Either correct the ddname if it is misspelled in the DD statement or the ddlist, or insert a new DD statement with this name.

The IEBGENER Program

IEB303I INVALID CONTROL STATEMENT.

Explanation: The construction of the above control statement is invalid.

System Action: The job is terminated. (The return code is 12.)

IEB304I CONTROL STATEMENT INPUT ERROR.

Explanation: A permanent error was detected while reading the SYSIN data set.

System Action: The job is terminated. (The return code is 12.)

IEB305I JOB TERMINATED AFTER LABEL EXIT.

Explanation: The return code from a label exit routine indicates termination.

System Action: The job is terminated. (The return code is 16.)

IEB306I JOB TERMINATED AFTER KEY EXIT.

Explanation: The return code from a KEY exit routine indicates termination.

System Action: The job is terminated. (The return code is 12 or 16, determined by the exit routine.)

IEB307I JOB TERMINATED AFTER DATA EXIT.

Explanation: The return code from a DATA exit routine indicates termination.

System Action: The job is terminated. (The return code is 12 or 16, determined by the exit routine.)

IEB308I PERMANENT INPUT ERROR.

Explanation: A permanent error was detected while reading the SYSUT1 data set.

System Action: The job is terminated. (The return code is 12.)

IEB309I PERMANENT OUTPUT ERROR.

Explanation: A permanent error was detected while writing the SYSUT2 data set.

System Action: The job is terminated. (The return code is 12.)

IEB310I DIRECTORY WRITE ERROR.

Explanation: A permanent error was detected while writing the directory of the SYSUT2 data set. This error could result if the SYSUT2 data set is not partitioned.

System Action: The job is terminated. (The return code is 12.)

IEB311I OUTPUT DIRECTORY FILLED.

Explanation: The directory of the SYSUT2 data set does not contain sufficient space for all the member entries.

System Action: The job is terminated. (The return code is 12.)

IEB312I JOB TERMINATED AFTER ERROR EXIT.

Explanation: The return code from an ERROR exit routine indicated termination.

System Action: The job is terminated. (The return code is 12.)

IEB315I SPACE NOT AVAILABLE.

Explanation: The required main storage space is not available.

System Action: The job is terminated. (The return code is 12.)

IEB316I DDNAME xxx CANNOT BE OPENED

Explanation: The named DD statement does not exist.

System Action: The job is terminated. (The return code is 12.)

User Response: Either correct the ddname if it is misspelled in the DD statement or the ddlist, or insert a new DD statement with this name.

The IEBPTPCH Program

IEB401I PRINT/PUNCH STATEMENT NOT FIRST.

Explanation: A PRINT or PUNCH statement is not the first utility control statement.

System Action: The job is terminated. (The return code is 12.)

IEB402I INVALID OPERATION, JOB TERMINATED.

Explanation: The operation in the above utility statement is invalid.

System Action: The job is terminated. (The return code is 12.)

IEB403I MORE THAN TWO TITLE STATEMENTS.

Explanation: More than two TITLE statements are included.

System Action: The job is terminated. (The return code is 12.)

IEB404I KEYWORD INVALID OR OMITTED.

Explanation: A required keyword is either incorrectly written or not included in the above statement.

System Action: The job is terminated. (The return code is 12.)

IEB405I PARAMETER INVALID OR OMITTED.

Explanation: A required parameter is either incorrect, inconsistent, or not included in the above statement.

System Action: The job is terminated. (The return code is 12.)

IEB406I JOB TERMINATED AFTER USER EXIT.

Explanation: The job was terminated after control was returned from an exit routine.

System Action: The job is terminated. (The return code is 12 or 16, determined by exit routine.)

IEB407I JOB TERMINATED DUE TO I/O ERROR.

Explanation: A permanent error was encountered.

System Action: The job is terminated. (The return code is 12.)

IEB408I MEMBER xxx CANNOT BE FOUND.

Explanation: The specified member is not contained in the SYSUT1 data set.

System Action: The member was not printed/punched. (The return code is 8.)

IEB409I INVALID CONTROL STATEMENT

Explanation: The construction of the above control statement is invalid.

System Action: The job is terminated. (The return code is 12.)

IEB410I INCORRECT RECORD STATEMENT

Explanation: The above RECORD statement is written incorrectly.

System Action: The job is terminated. (The return code is 12.)

IEB411I DDNAME xxx CANNOT BE OPENED

Explanation: The named DD statement does not exist.

System Action: The job is terminated. (The return code is 12.)

User Response: Either correct the ddname if it is misspelled in the DD statement or the ddlist, or insert a new DD statement with this name.

The IEBUPDAT Program

IEB501I INVALID EXIT NAME. JOB TERMINATED.

Explanation: An exit routine name in the EXEC statement is invalid.

System Action: The job is terminated. (The return code is 12.)

IEB502I EXIT RETURN CODE INDICATES TERMINATION.

Explanation: The return code from an exit routine is 16.

System Action: The job is terminated.

IEB503I I/O ERROR ON SYSUT1. JOB TERMINATED.

Explanation: A permanent error was encountered while the SYSUT1 data set was being read.

System Action: The job is terminated. (The return code is 12.)

IEB504I I/O ERROR ON SYSIN. JOB TERMINATED.

Explanation: A permanent error was encountered while the SYSIN data set was being read.

System Action: The job is terminated. (The return code is 12.)

IEB505I I/O ERROR ON SYSUT2. JOB TERMINATED.

Explanation: A permanent error was encountered while the SYSUT2 data set was being written.

System Action: The job is terminated. (The return code is 12.)

IEB509I CURRENT TRANSACTION REJECTED.

Explanation: The transaction represented by the printed control statement and logical record statements is rejected because the control statement is written incorrectly appears in the wrong position with respect to other control statements.

System Action: Processing continues with the next member of the library. (The return code is 4.)

IEB510I NO RECORDS WITHIN DELETE RANGE.

Explanation: No records were found within the range specified in the DELET statement.

System Action: Processing continues with the next member of the library. (The return code is 4.)

IEB511I NO RECORDS WITHIN NUMBER RANGE.

Explanation: No records were found within the range specified in the NUMBR statement.

System Action: Processing continues with the next member of the library. (The return code is 4.)

IEB512I DIRECTORY WRITE ERROR.

Explanation: A permanent error was detected while writing the directory of the SYSUT2 data set. This error could result if the SYSUT2 data set is not partitioned.

System Action: The job is terminated. (The return code is 16.)

IEB513I OUTPUT DIRECTORY FILLED.

Explanation: The directory of the SYSUT2 data set does not contain sufficient space for all the member entries.

System Action: The job is terminated. (The return code is 12.)

IEB514I MEMBER HAS NO RECORDS.

Explanation: The member identified in the printed header statement contains no records.

System Action: Processing continues with the next member of the library. (The return code is 4.)

IEB515I IMPROPER INVOCATION PARAMETER.

Explanation: Either the program or the EXEC statement calling IEBUPDAT has incorrectly passed parameters.

System Action: The request is terminated. (The return code is 12.)

IEB516I MEMBER NAME SEQUENCE ERROR.

Explanation: Member names, specified on header statements, are not in binary collating sequence.

System Action: Processing continues with the next member of the library. (The return code is 4.)

IEB517I DDNAME xxx CANNOT BE OPENED

Explanation: The named DD statement does not exist.

System Action: The job is terminated. (The return code is 12.)

User Response: Either correct the ddname if it is misspelled in the DD statement or the ddlist, or insert a new DD statement with this name.

This appendix, message appendixes in other publications, and the publication IBM System/360 Operating System: Control Program Messages and Completion Codes are designed so that the user can select the messages applicable to his installation and incorporate them in a binder.

Independent utility programs write error messages and diagnostic messages. Error messages describe error conditions associated with the utility programs. Diagnostic messages describe and locate faulty conditions associated with the hardware.

Error Messages

Error messages are listed in alphameric order with explanations and, when applicable, a recommended response.

IBC101W INVALID CARD CODE. JOB TERMINATED.

Explanation: An invalid card code appears in the above card.

IBC102A CONTROL STATEMENT ERROR. JOB TERMINATED.

Explanation: A utility control statement contains an incorrect keyword, parameter, or name field.

IBC103A STATEMENT SEQUENCE ERROR. JOB TERMINATED.

Explanation: The utility statements are not in the proper sequence, or unnecessary utility statements are present.

IBC104W SVC INTERRUPT. JOB TERMINATED.

Explanation: An SVC was initiated without a response having been defined.

IBC105A DEFINE INPUT DEVICE.

Response: Enter the following message from console typewriter: INPUT=dddd cuu, where dddd is the device type and cuu is the channel and unit address of the input device.

IBC106A THE VOLID IN CONTROL STATEMENT DOES NOT AGREE WITH ID IN VOL LABEL WHICH FOLLOWS. VOLID=xxx.

Explanation: The VOLID parameter in the utility control statement did not match the volume serial number found on the receiving volume (xxx).

Response: Correct statement or mount correct volume and restart program.

IBC107W TRACK ZERO BAD. JOB TERMINATED.

Explanation: The device cannot be initialized as a systems residence volume due to a defective surface on cylinder 00, track 00.

IBC108A HA or R0 FIELD BAD. JOB TERMINATED.

Explanation: The device cannot be initialized due to a bad surface area in the home address or track descriptor record areas.

IBC109I TRACK CHK INDICATES TRACK IS GOOD.

Explanation: The track in question is good and no alternate was assigned.

IBC110I BAD TRACK ccchhhh.

Explanation: A defective track was found at the specified location (cccc is the cylinder number, hhhh is the head number).

IBC111I ALTERNATE ccchhhh.

Explanation: An alternate track at

the specified location is assigned to replace the defective track (cccc is the cylinder number, hhhh is the head number).

IBC112W ALT TRACKS DEPLETED. JOB TERMINATED.

Explanation: The number of alternate tracks assigned has exceeded the maximum number for this device.

IBC151W MACHINE CHECK. JOB TERMINATED.

Explanation: A machine malfunction has caused a machine interrupt resulting in termination of the job.

IBC152W PROGRAM INTERRUPT. JOB TERMINATED.

Explanation: A program interrupt has occurred resulting in termination of the job.

IBC153A TYPEWRITER FAILED TO READ LAST MESSAGE. DEPRESS INTERRUPT KEY.

Explanation: The console typewriter failed to read the input message.

Response: Depress the console interrupt key and attempt to enter the input message again.

IBC154A READY READER cuu. DEPRESS INTERRUPT KEY.

Explanation: The reader has a card jam, a transport jam, or is out of cards.

Response: Correct the faulty condition and depress the console interrupt key to continue the program.

IBC155A READY PRINTER cuu. DEPRESS INTERRUPT KEY.

Explanation: The printer is not ready due to a forms check, an open interlock, or a depressed stop key.

Response: Correct the faulty condition and depress the console interrupt key to continue the program.

IBC156A READY TAPE cuu. DEPRESS INTERRUPT KEY.

Explanation: The tape drive on channel c, unit uu is not ready.

Response: Correct the faulty condition and depress the console interrupt key.

IBC157A READY DISK cuu. DEPRESS INTERRUPT KEY.

Explanation: The disk drive on channel c, unit uu is not ready.

Response: Correct the faulty condition and depress the console interrupt key.

IBC158A WRONG TAPE ON cuu. MOUNT PROPER TAPE. INTERRUPT.

Explanation: The tape on the specified device does not pertain to this job.

Response: Mount the correct tape and depress the interrupt key.

IBC159A READER CHECK. CORRECT ERROR. DEPRESS INTERRUPT KEY.

Explanation: A reader check has occurred.

Response: Correct the faulty condition and clear the reader check. Continue the program by depressing the console interrupt key.

IBC160A PRINT CHECK, CORRECT ERROR. DEPRESS INTERRUPT KEY.

Explanation: A print check has occurred.

Response: Correct the faulty condition and clear the print check. Depress the console interrupt key to continue the program.

IBC161A END OF TAPE. MOUNT TAPE ON cuu. DEPRESS INTERRUPT KEY.

Explanation: End of present tape reel.

Response: Mount another tape volume on the active tape device, i.e., the TODEV device for DUMP operations or the FROMDEV device for RESTORE operations.

IBC162A MOUNT ANOTHER PACK ON UNIT cuu. DEPRESS INTERRUPT KEY.

Explanation: End of the present disk pack.

Response: Mount another disk pack on the active disk drive, i.e., the TODEV device for DUMP operations or the FROMDEV device for RESTORE operations.

Diagnostic Messages

Diagnostic messages appear in the following format:

number (16-byte text)cuuxxssssyyyy

where c is the channel of the device in error, uu is the unit, xx is the command code, ssss are the status bytes from the channel status word, and yyyy are the sense bytes.

The following message texts are listed in alphameric order. All of the messages except IBC202A describe conditions that cause termination of the job.

IBC201W COMMAND REJECT.

Explanation: The specified channel has rejected an incorrect channel command word (CCW) list.

IBC202A INTERV. REQUIRED.

Explanation: The specified device is not ready.

Response: The specified device requires operator intervention to make it ready.

IBC203W BUS. OUT CHECK.

Explanation: A bus out check has occurred on the specified channel.

IBC204W EQUIPMENT CHECK.

Explanation: An equipment failure has occurred.

IBC205W DATA CHECK.

Explanation: A solid data check has occurred on the specified device.

IBC206W OVERRUN.

Explanation: An overrun check has occurred on the specified channel.

IBC207W FLAGGED TRACK.

Explanation: A track condition check has occurred on the specified device.

IBC208W DATA CONV. CHECK.

Explanation: A data converter check has occurred on the specified device.

IBC209W END OF CYLINDER.

Explanation: An unusual end of cylinder condition has occurred on the specified device.

IBC210W INVALID ADDRESS.

Explanation: An invalid address has been issued to the specified device.

IBC211W NOT AVAILABLE.

Explanation: The specified device is not attached to the system.

IBC212W READ DATA CHECK.

Explanation: A permanent read data check has been detected on the specified tape unit.

IBC213W COUNT FIELD CHECK.

Explanation: A data check has occurred in the count field of the specified direct-access device.

IBC214W TRACK OVERRUN.

Explanation: A track overrun condition has occurred.

IBC215W FILE PROTECTED.

Explanation: The specified device is file protected.

IBC216W DASD-END OF FILE.

Explanation: An unusual end of file has occurred on the specified direct-access storage device.

IBC217W NO RECORD FOUND.

Explanation: A no record found condition has occurred on the specified direct-access device.

IBC218W INVALID ERROR.

Explanation: An invalid error return has occurred.

IBC219W WRONG ERROR.

Explanation: The error return is valid but is not associated with the specified device.

IBC220W CHAN. CTRL ERROR.

Explanation: A channel control check has occurred on the specified channel.

IBC221W INTERFACE ERROR.

Explanation: An interface control check has occurred on the specified channel.

IBC222W CHAN. DATA CHECK.

Explanation: A channel data check has occurred on the specified channel.

IBC223W DASD OVERFLOW.

Explanation: An overflow incomplete condition has occurred on the specified direct-access device.

IBC224W PROGRAM CHECK.

Explanation: A program check has occurred due to an incorrect channel command word (CCW).

IBC225W PROTECTION CHECK.

Explanation: A protection check has occurred on the specified device.

IBC226W UNIT EXCEPTION.

Explanation: A unit exception has occurred on the specified unit.

IBC227W INCORRECT LENGTH.

Explanation: A wrong length record condition has occurred on the specified unit.

IBC228W CHAINING CHECK.

Explanation: A chaining check has occurred on the specified channel.

IBC229W COMMAND SEQ. ERR.

Explanation: An invalid sequence of channel command words (CCWs) was issued.

IBC230W SEEK CHECK ERROR.

Explanation: An invalid SEEK address was issued, or a unit malfunction caused a SEEK check.

IBC231W WRITE DATA CHECK.

Explanation: A permanent write data check has occurred on the specified tape unit.

IBC232W TAPE -- LOAD POINT.

Explanation: A tape at load point condition has occurred on the specified tape unit.

IBC233W NOISE RECORD.

Explanation: A noise record was found on the specified tape unit.

IBC234W MISSING ADR-MARK.

Explanation: A missing address marker has occurred on the specified device.

IBC249W I/O ERROR, JOB TERMINATED.

Explanation: This message follows all messages that describe input/output error conditions.

Printed in U.S.A. C28-6586-2

IBM

International Business Machines Corporation
Data Processing Division
112 East Post Road, White Plains, New York 10601

READER'S COMMENTS

Title: IBM System/360 Operating System Utilities

Form: C28-6586-2

Is the material:	Yes	No
Easy to Read?	___	___
Well organized?	___	___
Complete?	___	___
Well illustrated?	___	___
Accurate?	___	___
Suitable for its intended audience?	___	___

How did you use this publication?

___ As an introduction to the subject ___ For additional knowledge

Other _____

fold

Please check the items that describe your position:

___ Customer personnel ___ Operator ___ Sales Representative

___ IBM personnel ___ Programmer ___ Systems Engineer

___ Manager ___ Customer Engineer ___ Trainee

___ Systems Analyst ___ Instructor Other_____

Please check specific criticism(s), give page number(s),and explain below:

___ Clarification on page(s)

___ Addition on page(s)

___ Deletion on page(s)

___ Error on page(s)

Explanation:

CUT ALONG LINE

fold

Name _____

Address _____

FOLD ON TWO LINES,STAPLE AND MAIL
No Postage Necessary if Mailed in U.S.A.

C28-6586-2

staple

fold fold

FIRST CLASS
PERMIT NO. 81

POUGHKEEPSIE, N.Y.

BUSINESS REPLY MAIL
NO POSTAGE STAMP NECESSARY IF MAILED IN U.S.A.

POSTAGE WILL BE PAID BY

IBM CORPORATION
P.O. BOX 390
POUGHKEEPSIE, N. Y. 12602

ATTN: PROGRAMMING SYSTEMS PUBLICATIONS
DEPT. D58

fold

Printed in U.S.A. C28-6586-2

IBM

International Business Machines Corporation
Data Processing Division
112 East Post Road, White Plains, New York 10601

staple